

Eduardo de Almeida

**Implementação remota de uma *dashboard*
interativa para exibição de métricas simultâneas
dos *host* de um *cluster* do tipo *Beowulf***

São Carlos

2020

Eduardo de Almeida

**Implementação remota de uma *dashboard* interativa para
exibição de métricas simultâneas dos *host* de um *cluster*
do tipo *Beowulf***

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da Uni-
versidade de São Paulo como requisito para
obtenção do título de Engenheiro Eletricista.
Curso de Engenharia Elétrica com ênfase em
eletrônica

Orientador: Professor Doutor Carlos Dias Maciel

São Carlos

2020

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

DA447I
i de Almeida, Eduardo
Implementação remota de uma dashboard interativa
para exibição de métricas simultâneas dos hosts de um
cluster do tipo Beowulf / Eduardo de Almeida;
orientador Carlos Dias Maciel. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com
ênfase em Eletrônica) -- Escola de Engenharia de São
Carlos da Universidade de São Paulo, 2020.

1. Beowulf Cluster. 2. Netdata dashboard. 3. Redes
de Computadores. 4. Gerenciamento cluster. 5.
Configuração remota. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Eduardo de Almeida

Título: “Implementação remota de uma dashboard interativa para exibição de métricas simultâneas dos host de um cluster do tipo Beowulf”

Trabalho de Conclusão de Curso defendido e aprovado
em 04./12./2020

com NOTA 10,0 (Dez. , Zero), pela Comissão Julgadora:

Prof. Associado Carlos Dias Maciel - Orientador - SEL/EESC/USP

Mestre Jordão Natal de Oliveira Júnior - Doutorando -
SEL/EESC/USP

Mestre Matheus de Souza Sant'Anna Fogliatto - Doutorando -
SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Agradecimentos

Gostaria de agradecer, primeiramente ao orientador, professor Doutor Carlos Dias Maciel pela oportunidade de trabalhar com tópicos nessa área. Gostaria também de agradecer ao meus pais e familiares pelo suporte e apoio incondicional.

Resumo

O sistema de *Cluster* do tipo *Beowulf* é formado com computadores básicos, partilhando processos paralelos entre eles, sendo assim um *Cluster* apresenta escalabilidade que pode ser empregada em *data mining*, processamento de dados, servidores de *Internet*, *back-end*. Em cada um destes computadores é instalada uma distribuição de *linux* com *software aberto* sob uma topologia de rede *LAN*, ligada em uma rede *mesh swicht* com *fastethernet*. Enquanto o *Beowulf* é utilizado para desenvolver programação paralela deve ser monitorizado de modo a manter a eficiência da infraestrutura permitindo assim resolução de problemas no sistema de *cluster*. Ora, monitoramento é fundamental não só para as operações de *software* mas também para otimizar o sistema e preveni-lo de danos permanentes ao *hardware* evitando sobrecarregar nós individuais do *cluster*. Destaca-se o ganho de robustez nas diretivas de resolução de problemas no sistema de *cluster*, uma vez que por meio da periodicidade métrica o sistema é avaliado globalmente de forma quantitativa. Dessa forma, além de manter a saúde do sistema o monitoramento garante o nível e desempenho dos serviços e aplicações, mantendo os dados armazenados para detectar e corrigir falhas, técnica conhecida como (*Troubleshooting*). A implementação da *dashboard* e a configuração da ferramenta de *streaming* do *software* requerem a priori a identificação dos parâmetros da rede, o que é feito utilizando ferramentas abertas de mapeamento de rede disponíveis em *linux*. Assim, um cliente remoto pode gerir um sistema de *clustering* por meio da ferramenta *OpenSSH*, utilizando-se do protocolo *SSH* e da implementação de *tunneling* para encaminhamento de portas. Este trabalho é motivado pelo monitoramento em tempo real de métricas customizáveis, como utilização de memória, uso *CPU*, velocidade *fan*, largura de banda da rede, taxa de gravação em disco e etc. Exibem-se esses parâmetros previamente configurados em uma *dashboard* de forma remotamente, isto é, sem estar conectador á rede interna local. **Palavras-chave:** *Beowulf Cluster*. *Netdata dashboard*. Configuração remota. Gerenciamento cluster. Redes de Computadores.

Abstract

Beowulf Clustering is formed with regular computers, sharing paralleling process among them, thus are scalable performance cluster wich can be ussed as data mining, data processing, internet server. In each of these computer are installed a open software linux distribuiton arranged into a dedicate lan area network, coneccted with a mesh swicht with fastethernet . While Beowulf are used to develop parallel programming it has to be monitored to maintain efficient infrastructure and troubleshooting in cluster system. Thus, monitoring is fundamental not only to software operations but to optimize system. The implementation of monitoring dashboard and configuration of software streaming tool require the identification of the network parameters, wich is done using linuxs networking mapping open tools. Thus a remote SSH client ussing settings, like tunneling and port fowarding can manage cluster. This work is motivated by monitoring real-time metrics of memory usage, cpu thermal, fan speed, networking bandwidth, disk write rate, remotely from outside of localhost network. display those actions in a same monitoring and by troubleshooting directives thus dashboard can be used for watching entire systems, maintaining health and performance of systems services and applications.

Keywords: Beowulf Cluster. Netdata dashboard. Remote Configure. Cluster management.

Lista de ilustrações

Figura 1 – Topologia de uma rede comutada	33
Figura 2 – Bandas de frequência atualmente disponíveis e bandas de frequência atribuídas previstas para as principais tecnologias sem fios	35
Figura 3 – Endereçamento <i>IPv4</i> de classe A (a), B (b) e C (c)	38
Figura 4 – Exemplificação de endereçamento <i>IPv4</i> de classe C (c)	42
Figura 5 – Etapas de conexão <i>SSH</i>	51
Figura 6 – Especificações do <i>sshd_config</i>	55
Figura 7 – Autenticação cliente-servidor na conexão <i>SSH</i>	56
Figura 8 – Especificações do <i>sshd_config</i>	58
Figura 9 – Especificações do <i>Proxy</i> no Sistema Operacional	61
Figura 10 – Diagrama de blocos de uma estrutura canônica de <i>cluster</i>	64
Figura 11 – Topologia <i>streaming netdata</i>	71
Figura 12 – <i>Dashboard</i> nativa do <i>software netdata</i>	74
Figura 13 – Pacotes de dependências <i>netdata</i>	80
Figura 14 – <i>Script</i> de instalação <i>netdata</i>	81
Figura 15 – Diretório dos arquivos <i>netdata</i>	83
Figura 16 – Saída do instalador <i>netdata</i>	84
Figura 17 – Detalhamento interface de rede	86
Figura 18 – Mapeamento de portas	88
Figura 19 – Alteração no campo <i>API</i> do <i>stream.conf</i>	92
Figura 20 – Alteração no campo <i>MGUID</i> do <i>stream.conf</i> nó principal	93
Figura 21 – Alteração no nó secundário do <i>stream.conf</i>	95
Figura 22 – <i>Script</i> de alteração nós secundários	98
Figura 23 – Acesso á <i>Dashboard</i> principal	102
Figura 24 – <i>Mirrored_host</i>	103
Figura 25 – Acesso a todas <i>Dashboard</i>	105
Figura 26 – <i>Multi-Dashboard</i> para todas máquinas	107

Lista de tabelas

Tabela 1	–	Faixa de endereçamento <i>IPv4</i> .	38
Tabela 2	–	Endereços especiais <i>IPv4</i> reservados para <i>LAN</i> .	40
Tabela 3	–	Possíveis configurações dos nós no <i>netdata</i> .	70

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
ARP	<i>Address Resolution Protocol</i>
BASH	<i>Bourne Again Shell</i>
CLI	<i>Command Line Interface</i>
DHCP	<i>Dynamic Host Configuration Protocol</i>
DMZ	<i>Demilitarized Zone</i>
DNS	<i>Domain Name Service</i>
(EIA)	<i>Electronic Industries Association</i>
TIA	<i>Telecommunications Industry Association</i>
CIDR	<i>Classless Inter-Domain Routing</i>
CLI	<i>Command Line Interface</i>
CSMA/CA	<i>Carrier Sense Multiple Access with Collision Avoidance</i>
EPEL	<i>Extra Packages for Enterprise Linux</i>
ECDSA	<i>Elliptic Curve Digital Signature Algorithm</i>
FHS	<i>Filesystem Hierarchy Standard</i>
FS	<i>File System</i>
FTP	<i>File Transfer Protocol</i>
HD	<i>Hard Drive</i>
HPC	<i>High-performance computing</i>
HTTP	<i>Hypertext Transfer Protocol</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
IAM	<i>Identity Access Management</i>
IEE	<i>Institute of Electrical and Electronics Engineers</i>

IETF	<i>Internet Enginnering Task Force</i>
IoT	<i>Internet Of Things</i>
IP	<i>Internet Protocol</i>
ICANN	<i>Internet Corporation for Assigned Names and Numbers</i>
IPTV	<i>Internet Protocol Television</i>
IPv4	<i>Internet Protocol Version 4</i>
IPv6	<i>Internet Protocol Version 6</i>
ISI	<i>Intersymbol interference</i>
ISM	<i>Industrial Scientific and Medical networks</i>
ISOC	<i>Internet Society</i>
JS	<i>JavaScript</i>
JSON	<i>JavaScript Object Notation</i>
KVM	<i>KernelVirtual Machine</i>
KSM	<i>Kernel Samepage Merging</i>
LAN	<i>Local Area Network</i>
LDAP	<i>Lightweight Directory Access Protocol</i>
LPS	<i>Laboratório de Processamento de Sinais</i>
LSB	<i>Linux Standard Base</i>
LDAP	<i>Lightweight Directory Access Protocol</i>
MAC	<i>Media Acess Control</i>
MIC	<i>Market Intelligence & Consulting Institute</i>
MIMO	<i>Multiple Input Multiple Output</i>
MPI	<i>Message Passing Interface</i>
MPIH	<i>Message Passing Interface Highperformace</i>
NAT	<i>Network Address Translation</i>
NFS	<i>Network File System</i>

NIC	<i>Network Interface controller</i>
NMAP	<i>Network Mapper</i>
OFDM	<i>Orthogonal Frequency-Division Multiplexing</i>
OpenVPN	<i>Open Source Connection Protocol</i>
OSI	<i>Open Systems Interconnection</i>
PDU	<i>Protocol Data Unit</i>
POSIX	<i>Portable Operating System Interface</i>
PKCS	<i>Public Key Cryptography Standard</i>
OpenPGP	<i>Open Pretty Good Privacy</i>
OpenSCP	<i>Open Source Connection Protocol</i>
OpenVPN	<i>Open Virtual Private Network</i>
PMS	<i>Package Management System</i>
POP3	<i>Post Office Protocol version 3</i>
RFC	<i>Request for Comments</i>
RAM	<i>Random Access Memory</i>
RPM	<i>Red Hat Package Manager</i>
SCMS	<i>SMILE Cluster Management</i>
SCP	<i>Secure Copy</i>
SNIA	<i>Storage Networking Industry Association</i>
SOCKS5	<i>Socket Secure</i>
SH	<i>Shell</i>
SSH	<i>Secure Shell</i>
SSHD	<i>SSH Daemon (SSHD)</i>
SSH-CONN	<i>SSH Connection Protocol</i>
SSHFS	<i>SSH Filesystem</i>
SSL	<i>Secure Sockets Layer</i>

SIG	<i>Special Interest Groups</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
RSA	<i>Rivest-Shamir-Adleman</i>
TCP	<i>Transmission Control Protocol</i>
TCP/IP	<i>Transmission Control Protocol/Internet Protocol</i>
UDP	<i>User Datagram Protocol</i>
UHF	<i>Ultra High Frequency</i>
URL	<i>Uniform Resource Locator</i>
UTP	<i>Unshield Twisted Pair</i>
VIF	<i>Virtual Interface</i>
WAN	<i>Wireless Area Network</i>
WI-FI	<i>Wireless Fidelity</i>
WLAN	<i>Wireless Local Area Network</i>

Sumário

1	INTRODUÇÃO	23
1.1	Objetivo	26
1.2	Estrutura do texto	28
2	TEORIA	31
2.1	Redes	31
2.1.1	Conectividade	31
2.1.1.1	<i>Wi-Fi</i>	36
2.1.1.2	802.11n	36
2.1.2	Endereçamento	37
2.1.2.1	Endereçamento <i>IPv4</i>	37
2.2	<i>Linux</i>	44
2.2.1	<i>Distribuições</i>	44
2.2.2	Estrutura	45
2.2.3	Comandos	47
2.2.4	Permissões	49
2.2.5	<i>SSH</i>	50
2.2.5.1	Utilitários instalados com <i>OpenSSH</i>	52
2.2.5.2	Configuração <i>OpenSSH</i>	54
2.2.6	Multi servidor <i>OpenSSH</i>	57
2.2.7	<i>SSH Tunnel</i>	59
2.2.8	Pacotes e dependências	61
2.3	<i>Cluster</i>	63
2.4	<i>Software Netdata</i>	64
2.4.0.1	<i>Database</i>	66
2.4.1	<i>Streaming</i>	70
2.4.2	<i>API</i>	72
2.4.3	<i>Dashboard</i>	72
3	MATERIAIS E MÉTODOS	77
3.1	Materiais	77
3.2	Métodos	78
3.2.1	Acesso ao <i>cluster</i>	78
3.2.2	Instalação <i>netdata</i>	79
3.2.2.1	Instalação dependências <i>netdata</i>	79
3.2.2.2	Instalação efetiva do <i>netdata</i>	80

3.2.2.3	Comandos para gerenciamento <i>netdata</i>	81
3.2.2.4	Diretórios <i>netdata</i>	82
3.2.2.5	Instalação automatizada	83
3.2.2.6	Conclusão do processo instalação	84
3.2.3	Configuração <i>netdata</i>	84
3.2.3.1	Mapeamento da rede	85
3.2.3.2	Configuração <i>streaming</i>	88
3.2.3.2.1	Configuração <i>streaming</i> no nó principal	90
3.2.3.2.2	Nós secundários - <i>child</i>	93
3.2.4	Configuração da <i>dashboard</i> no <i>netdata</i>	96
4	RESULTADOS	101
4.1	Resultados <i>Dashboard</i> principal	101
4.1.1	Resultados <i>Streaming</i>	103
4.1.2	Resultados <i>Multi-Dashboard</i>	106
5	CONCLUSÃO	109
	REFERÊNCIAS	111
	APÊNDICE A – <i>SCRIPT</i> DE INSTALAÇÃO DO PACOTE DE DEPENDÊNCIAS DO <i>NETDATA</i>	123
	APÊNDICE B – <i>SCRIPT GIT</i> PARA COPIAR DIRETÓRIO DO <i>NETDATA</i>	125
	APÊNDICE C – <i>SCRIPT</i> DE INICIALIZAÇÃO AUTOMÁTICA . .	127
	APÊNDICE D – SINTAXE PARA ADICIONAR <i>MGUID</i> NO <i>PROXY</i>	129
	APÊNDICE E – ALTERAÇÕES NOS ARQUIVOS <i>.CONF</i> PARA O NÓ PRINCIPAL	131
	APÊNDICE F – ALTERAÇÕES NOS ARQUIVOS <i>.CONF</i> PARA OS NÓS SECUNDÁRIOS	133
	APÊNDICE G – <i>SCRIPT</i> DE ALTERAÇÃO NÓS SECUNDÁRIOS	135
	APÊNDICE H – EXEMPLO DE MODIFICAÇÃO NA <i>DASHBOARD</i> PRINCIPAL	137

APÊNDICE I – ALTERAÇÕES NO TRECHO DO ARQUIVO <i>.HTML</i> PARA O NÓ PRINCIPAL	139
APÊNDICE J – A SINTAXE <i>DASH-*</i> EXEMPLIFICADA	141

1 Introdução

As simulações matemáticas usando computação de alto desempenho são ferramentas essenciais para o desenvolvimento de diversas aplicabilidades científicas (FABRICIUS et al., 2005). Consequentemente, os custos de computação de alto desempenho constituem um componente essencial dentro dos programas de graduação e pós-graduação. Diante desse cenário surge a técnica de *clustering* de modo a apresentar uma ferramenta mais barata e eficiente para processamentos computacionais elevados, tendo se popularizado ultimamente devido ao seu alto desempenho proveniente da paralelização de processamento (DATTI; UMAR; GALADANCI, 2015).

Clustering é uma técnica para estender as capacidade de uma classe de componentes existente, de modo a se obter uma maior capacidade computacional, utilizando-se principal de técnicas de computação paralela e rede de computadores (NEGUS, 2020). O *cluster* do tipo *Beowulf* apresenta vantagem na relação preço/desempenho inclusive sob outras estruturas de *cluster* (STERLING DANIEL F. SAVARESE, 1999).

O *Beowulf* está intrinsecamente relacionado ao conceito de rede de computadores, utilizando-se dos protocolos e arquiteturas de redes (DATTI; UMAR; GALADANCI, 2015). Uma vez que para sua implementação é utilizado uma rede de nós, com cada nó sendo um computador pessoal, normalmente de baixo custo, um nó pode fornecer serviços e também pode acessar serviços fornecidos por outros nós através da rede (Kai Shen; Tao Yang; Lingkun Chu, 2002). Dessa forma, o nó é responsável por todas atividades e recursos computacionais associados com execução e aplicação de programas, suportando um ambiente software sofisticado (PETERSON, 2003). Dessa forma, infere-se que um nó no sistema de *cluster* permite a execução de instrução, armazenamento de informação temporária em alta velocidade, armazenamento de informação com alta capacidade, além de comunicação com o ambiente externo incluindo por exemplo outros nós (STERLING DANIEL F. SAVARESE, 1999).

No contexto de rede de computadores, serviços e aplicações são implementados por meio de protocolos, sendo os protocolo responsáveis pela sintaxe e semântica do processo de comunicação (KUROSE JAMES F.; ROSS, 2017). Em um sistema de computação paralela os serviços são particionados, replicados e agregados de acordo com as requisições, tal modularização é obtida por meio de protocolos como *Message Passing Interface (MPI)* e *Parallel Virtual Machine (PVM)* (VERSTOEP et al., 2004), tratam-se de protocolo padronizados para passagem de mensagens aos processamentos de memória distribuído, amplamente utilizado em computação paralela (ADAMS; VOS, 2002a). Além disso, os serviços são disponibilizados aos *host* internamente na rede local utilizando-se de protocolos

de redes (PETERSON, 2003).

Dessa forma, apesar do processo de criação e gerenciamento de um cluster envolver conceitos de redes o sistema operacional também desempenha papel fundamental na técnica de *clustering* (STERLING DANIEL F. SAVARESE, 1999). Ora, uma aplicação se utiliza de protocolos integrados ao próprio sistema operacional para seu funcionamento (PETERSON, 2003). Para o funcionamento de um *cluster* é necessária a implementação e configuração de uma *Local Address Network (LAN)*, normalmente se baseando em arquiteturas de rede *FastEthernet* por meio de um *switch* e sendo endereçada e roteada pelo sistema operacional (HARRINGTON, 2007).

Além disso, são utilizados compiladores para execução de programas, tal como o *Message Passing Interface Highperformance (MPICH)*¹ uma implementação do protocolo (*MPI*) e disponível em sistemas operacionais do tipo *UNIX*² que permite implementação de multi-usuários e multi-tarefas (GROPP; THAKUR; BALAJI, 2020). Um *cluster* ainda se utiliza de protocolos de criptografia segura de rede como o *Secure-Shell (SSH)* para acesso remoto e de estruturação *Network File System (NFS)* para o compartilhamento do sistema de arquivos (Kai Shen; Tao Yang; Lingkun Chu, 2002), definida pela *Request for Comments (RFC)*³, tratam-se de uma publicação da *Internet Society (ISOC)*⁴ e associados, determinando técnicas de padronização desenvolvidas e mantidas pela instituição aberta que especifica voluntariamente os padrões de implementação global da internet, a *Internet Enginnering Task Force (IETF)*⁵ (NUECHTERLEIN, 2005).

Dessa forma, devido a característica de código livre bem como a existência de integração entre implementações de protocolos de processamento e de redes junto ao sistema operacional se justifica a instalação de um sistema *linux* em cada um dos nós, possibilitando o acesso, execução e configuração do *cluster* (STERLING DANIEL F. SAVARESE, 1999). Ora, a flexibilidade de um sistema de código aberto do tipo *UNIX* baseado no *kernel* do *linux*, permite facilmente modificações, reorganizações e ajuste para qualquer tarefa de modo eficiente (GROPP EWING LUSK, 2003), sendo o *kernel* uma integração entre *hardware* e *software* que fornece ao *hardware* um ambiente básico para processamento e gerenciamento de memória, realizando assim o gerenciamento de recursos do sistema (Wang et al., 2009).

A técnica de *clustering* pode ser implementada em diferentes distribuições do *linux*, sendo uma distribuição o *kernel* do *linux* empacotado com todas as ferramentas e componentes que funcionam juntamente a esse, assim um pacote de distribuição *linux* agrupa todos programas comuns e interfaces, como *desktop*, *web server*, compiladores entre

¹Disponível em <<https://www.mpich.org>>

²Disponível em <<http://www.unix.org>>

³Disponível em: <<https://tools.ietf.org/html/rfc1094>>

⁴Disponível em: <<https://www.internetsociety.org>>

⁵Disponível em: <<https://www.ietf.org/>>

outras ferramentas (GROPP EWING LUSK, 2003). Entre as distribuições de código aberto estão popularmente *Ubuntu*⁶, *Debian*⁷, *Fedora*⁸, *LinuxMint*⁹, *ElementaryOS*¹⁰, *Archlinux*¹¹, *CentOS*¹², *OpenBSD*¹³ (NEGUS, 2020). No ano de 2016, foi feito um levantamento com pelo *StackOverflow* com 56,033 programadores usuários de *linux* em 173 diferentes países de modo que desenvolvedores respondendo 45 perguntas sobre os diferentes tópicos da área mostrando uma porcentagem de utilização para distribuições em *desktop* de 56.7% *Ubuntu*, 8.8% *Debian*, 7.8% *LinuxMint*, 6.5% *Fedora* e 20.3% Outras distribuições. O levantamento completo abrange diferentes parâmetros tentando traçar o perfil dos utilizadores de *linux* e está disponível no *StackOverflow* (TECHNOLOGY-DESKTOP-OPERATING-SYSTEM, 2020).

Um cluster *Beowulf* é implementado utilizando-se das propriedades de código aberto do sistema *linux*, incluindo bibliotecas e funcionalidades intrínsecas ao ambiente *open sourcer* (STERLING DANIEL F. SAVARESE, 1999). O sistema de *cluster* pode ser visto como sendo constituído por pelo menos quatro componentes, dois componentes associados ao *hardware* e dois ao *software* (GROPP EWING LUSK, 2003). Os componentes de *hardware* são os nós propriamente ditos que realizam o trabalho computacional executando as aplicações, outro componente de *hardware* é a rede propriamente dita, que interconecta os nós para formar um único sistema (GROPP EWING LUSK, 2003). Enquanto que as componentes de *software* de um cluster são utilizadas para desenvolver programas e aplicação de programação paralela, bem como o ambiente de *software* para gerenciar os recursos do *cluster* (SRINIVAS; RADHAKRISHNA; RAO, 2014).

Desse modo, a construção ou identificação da rede é parte fundamental para o funcionamento de um *cluster* de alto desempenho (GROPP EWING LUSK, 2003). Ora, a rede fornece os meios para troca de dados entre os nós do *cluster*, coordenando suas operações por meio de protocolos e mecanismos de sincronização global (STERLING DANIEL F. SAVARESE, 1999). Assim, devem-se abordar conceitos e fundamentos de redes de computadores na implementação de uma técnica adequada de *clustering* (PETERSON, 2003).

Devido às suas características entregando alto desempenho com baixo custo o *cluster* do tipo *Beowulf* tem se popularizado na área de processamento de dados em âmbito acadêmico (ADAMS; VOS, 2002a). Dessa forma, ao executar aplicações com elevada exigência computacional o principal empecilho que surge se dá quanto á manutenção

⁶Disponível em <<https://ubuntu.com>>

⁷Disponível em <<https://www.debian.org>>

⁸Disponível em <<https://getfedora.org>>

⁹Disponível em <<https://linuxmint.com>>

¹⁰Disponível em <<https://elementary.io>>

¹¹Disponível em <<https://http://www.archlinux.org>>

¹²Disponível em <<http://www.centos.org>>

¹³Disponível em <<http://www.freebsd.org>>

e gerenciamento da saúde do *cluster Beowulf*, isto é, a disponibilização imediata de dados associados à saúde do sistema por meio de métricas associadas a parâmetros reais como uso de memória, *CPU* ou tráfego de rede (STERLING DANIEL F. SAVARESE, 1999). Além disso, a maioria das técnicas de monitoramento empregadas na literatura como o *Clusmon* (KENNINGTON, 2006a) ou *SMILE Cluster Management (SCMS)* (UTHAYOPAS; MANEESILP; INGONGNAM, 2000) monitoram apenas parâmetros relacionados com programação e equilíbrio de carga no *cluster* (KENNINGTON, 2006b), não possibilitando fácil customização ou adaptação para especificidade de um parâmetro crítico utilizado por uma aplicação. Ora, com o advento da *internet* e a popularização do gerenciamento remoto de sistema é possível utilizar ou adaptar ferramentas que solucionam essa problemática (LEARN..., 2020).

Idealmente, deve-se possuir um indicador instantâneo sobre a saúde do sistema de modo a intervir manualmente diante de alguma anormalidade (LIPOR; BALZANO, 2020). Sendo assim imperativo que o administrador do sistema de *clustering* seja alertado em tempo real sobre eventual mal funcionamento do sistema e principalmente ser capaz de intervir remotamente (UTHAYOPAS; MANEESILP; INGONGNAM, 2000).

Visando solucionar essa dificuldade a configuração do software *Netdata* no *cluster* implementado no projeto tem como objetivo fornecer informativos sobre o desempenho do sistema em tempo real e com elevada granularidade, além de armazenar dados periodicamente usados como diretivas na solução de eventuais problemas (*troubleshooting*) (LEARN..., 2020). Quando implementada essa ferramenta permite a exibição de praticamente todos parâmetros de interesse de forma customizada, permitindo maior granularidade na detecção de anomalias e intervenção instantânea por meio do gerenciamento remoto de recursos (LEARN..., 2020).

No entanto, deve-se inicialmente mapear dados sobre a estruturação da rede interna e configurar os nós do *cluster* para que a ferramenta de monitoramento e o coletora de métricas seja executada de forma otimizada, mostrando ao usuário por meio de painéis interativos, chamados de *dashboard* o que efetivamente está sendo executado no sistema e de que forma os recursos computacionais estão sendo alocados (WEB..., 2020b). Tal interface de usuário é programável e os gráficos de métricas customizáveis, de modo que é possível especificar os diferentes tipos de gráficos para os parâmetros mostrados por meio de diferentes tipos de métricas e implementada por diferentes linguagens de programação (MULTI..., 2020).

1.1 Objetivo

O projeto tem como objetivo principal implementar de forma eficiente o monitoramento e análise de métricas de forma simultâneas em todos nós de um *cluster* do tipo

Beowulf no laboratório de processamento de sinais (LPS), utilizando-se de um *kernel* do *linux* compilado sob a distribuição *workstation fedora*¹⁴ de modo que todas as configurações possam ser efetuadas remotamente por meio de tunelamento do *Transmission Control Protocol/Internet Protocol (TCP/IP)* e encaminhamento de porta *SSH* permitindo acessar remotamente às máquinas de forma segura, instalando-se *packages* individualmente e configurando cada nó da rede para que haja comunicação com a *Application Programming Interface (API)* do software *Netdata* instalado e configurado manualmente em cada outro nó da rede do laboratório LPS (INSTALLER... , 2020). Com a devida configuração da rede é possível o monitoramento dos dados de todas as máquinas paralelamente e comparativamente, por meio de uma *dashboard* programável e customizável acessada remotamente a partir de qualquer rede de *internet* e por qualquer dispositivo com um *browser* instalado com suporte à *JavaScript*, adicionalmente é implementada uma *multi-dashboard* unificada para todos os nós do sistema também acessível por meio de tunelamento *SSH* (MULTI... , 2020).

Como os parâmetros do sistema são exclusivos para cada rede *LAN*, é necessário configurar a rede e cada nó devidamente, mesmo em uma topologia de rede devidamente estruturada é útil a configuração de *Internet Protocol (IP)* de forma estática internamente, de modo a viabilizar a comunicação de ferramentas do *software* instaladas nos diferentes nós (Amiri et al., 2012). Através do terminal *SSH* é possível configurar o endereçamento e roteamento estruturando a rede para as especificidades do *software* (NEGUS, 2020).

Dessa forma, devidamente configurado o *software* coletará métricas selecionadas, armazenando-as em uma base de dados interna individual para cada máquina que esteja se comunicando com outros nós da rede e executando o *Netdata*, de modo que será capaz de replicar comparativamente os dados em um painel, chamado *dashboard* (STREAMING... , 2020b). Ora, como o objetivo do projeto é realizar o monitoramento e análise simultaneamente de dados de todos os nós que compõem o cluster em tempo real, destaca-se a implementação e configuração da ferramenta *streaming* disponível com o *software* (STREAMING... , 2020a). No entanto, inicialmente é necessário implementar a *dashboard*, para tal é preciso instalar e configurar o agente *netdata*, isto é, o processo que executa a aplicação em plano de fundo no sistema operacional, referido como *daemon* em sistemas de computador multi tarefas, também recebendo como sinônimo a nomenclatura de serviço *daem*. Configura-se a *dashboard* de acordo as necessidades de monitoramento do laboratório por meio de *JavaScript (JS)* (WEB... , 2020b). Por fim, mostra-se ao usuário o resultado comparativamente das métricas resultantes de todos os nós da rede por meio de diferentes gráficos em um painel interativo criado exclusivamente para a topologia da rede do laboratório LPS (MULTI... , 2020).

¹⁴Disponível em <<https://getfedora.org>>

1.2 Estrutura do texto

No primeiro capítulo é abordada a teoria geral utilizada no desenvolvimento do projeto, iniciando-se pela teoria de redes de modo a explicar conceitos que são amplamente utilizados para o funcionamento de um *cluster* bem como para a implementação da ferramenta de *streaming* (STREAMING..., 2020a). Ora, sendo a técnica de *clustering* o agrupando de nós para gerir e executar várias tarefas de forma distribuída, deve-se ter um meio, ou canal para que ocorra comunicação (STERLING DANIEL F. SAVARESE, 1999). As diferentes técnicas de *clustering* propostas na literatura geralmente classificadas se baseiam no modelo global de arquitetura e operação da rede *LAN* com endereçamento *Internet Protocol version 4 (IPv4)* estático por meio de um *switch* e um roteador conectado à *internet* (ABBASI; YOUNIS, 2007).

Dessa forma, como a técnica de *clustering* está intrinsecamente relacionada ao conceito de redes e como uma rede de computador é responsável pelo compartilhamento dos recursos, isto é, um conjunto de nós conectados por um *link* de comunicação que efetivamente transmite a informação, então as métricas e dados do *cluster* são compartilhados na rede para acesso de todos os nós (STERLING DANIEL F. SAVARESE, 1999), de modo que os pacotes devem ser sincronizados ordenados e roteados (ABBASI; YOUNIS, 2007).

Na primeira sessão do capítulo um é abordado inicialmente o conceito de conectividade de modo a explicitar como acontece a comunicação em uma rede do tipo *LAN* e como se estabelece uma *Wireless Local Area Network (WLAN)*, aborda-se para tal qualitativamente as principais características da conexão *Wireless Fidelity (Wi-Fi)* amplamente utilizado na maioria das topologias de rede atuais (CHALLOO et al., 2012). O *Wi-Fi* é uma família de protocolo para rede sem fio, baseada na padronização 802.11 do *Institute of Electrical and Electronics Engineers (IEEE)*¹⁵ (GONG; ZHAO; YANG, 2014) e de acordo com um relatório global feito em 2018 pela *Research and Markets*¹⁶, referenciada como *Market Intelligence & Consulting Institute (MIC)* mais de 2,97 bilhões de dispositivos com *Wi-Fi* são comercializados todos os anos globalmente (WI-FI..., 2020).

Dessa forma, discorre-se inicialmente e brevemente sobre os sinais eletromagnéticos de modo a expor o aspecto do espectro eletromagnético em que são definidas as bandas de comunicação contemporâneas, incluindo a tecnologia *Wi-Fi* em que a rede *WLAN* do laboratório é estruturada.

Ainda nessa sessão são apresentados conceitos básicos de redes, de modo a explicitar a nomenclatura e significados envolvendo redes de computadores, descreve-se ainda de forma qualitativa como a conectividade acontece em diferentes níveis de comunicação (PETERSON, 2003). Em seguida, explora-se os conceitos de endereçamento e roteamento

¹⁵Disponível em <<https://www.ieee.org/802.11>>

¹⁶Disponível em <<https://www.researchandmarkets.com>>

de *IPv4* essenciais para a construção de uma rede *LAN* utilizada pelo *cluster* (D.E., 2000).

Na segunda sessão do capítulo são abordados conceitos associados ao *linux*, uma vez que o gerenciamento remoto de um *cluster* está intrinsecamente relacionado ao conhecimento de protocolos de comunicações e aplicações de comandos *linux* utilizando-se de comandos, pacotes de dependência, parâmetros de rede para acesso remoto por meio do *OpenSSH*.

Na terceira sessão do capítulo é descrito brevemente sobre conceitos associados ao *cluster* do tipo *Beowulf*. Ora, o objetivo do projeto é gerenciar remotamente o sistema por meio do sistema operacional linux propriamente dito implementando ferramenta coletora de métricas e parâmetros Na terceira sessão do capítulo são tratadas como as configurações remotas necessárias para o funcionamento do *software Netdata*.

2 Teoria

2.1 Redes

2.1.1 Conectividade

O conceito de conectividade está intrinsecamente relacionado á definição de rede, uma vez que uma rede é um conjunto de nós conectados por *links* de comunicação (PERLMAN, 1999). O processo de comunicação em uma rede deve fornecer conectividade entre computadores, trata-se assim de um processo complexo e por isso modularizado (KUROSE JAMES F.; ROSS, 2017). Ora, a conectividade se dá associada por protocolos atuando como sintaxe e semântica em diferente níveis definidos no modelamento teórico da camada *Open Systems Interconnection (OSI)* que por sua vez, são implementado na prática pela pilha de protocolos *TCP/IP* (WHITE, 2018). Como abordagem inicial geral, do ponto de vista da forma como os dados são compartilhados em uma rede são definidos inicialmente dois tipos básicos de rede quanto a comunicação externa, a rede ponto-a-ponto e cliente servidor (ABBASI; YOUNIS, 2007).

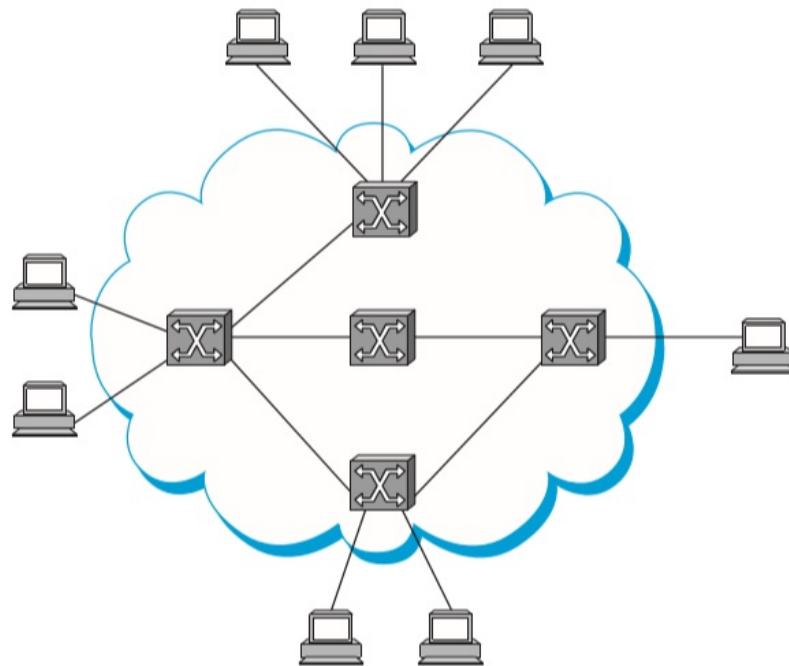
O exemplo mais simples consiste em dois computadores conectados por um meio físico, como um cabo par trançado, de modo que a ligação estabelecida é chamada de *link*, enquanto os computadores são chamados nesse caso de nós (*nodes*). Tal tipo de conexão é chamada de conexão ponto-a-ponto ou *Point-to-point links* (PETERSON, 2003). Nessa topologia não existe uma estruturação hierárquica nem a presença de um servidor, e por isso apresenta baixa segurança e escalabilidade, uma vez que necessita de cabeamento direto, por meio de um conector 8P8C modular (referido também como *Registered Jack-45 (RJ-45)* associado ao um cabeamento de cobre por par trançado (Cat), de modo a cancelar as interferências eletromagnéticas, referidas na literatura como *Electromagnetic interference (EMI)* e principalmente associadas ao efeito de *cross-talk* (KUROSE JAMES F.; ROSS, 2017). Ora, a transmissão do sinal está associado á uma corrente elétrica cuja propagação gera um campo magnético, de modo que caso os campos de fios adjacentes se sobrepujam eles interferem induzindo distorções nos sinais que podem ser interpretados erroneamente (HAYKIN, 2001). Dessa forma, o canal é conectado aos computadores individualmente por meio da placa de rede ou *Network Interface Controller (NIC)*, de modo que se assume que as fontes de informação são mutuamente independentes (AHLWEDE et al., 2000), no entanto nessa topologia o canal para transmissão é compartilhado e largura de banda é dividida caso outra *NIC* seja ativada simultaneamente (TANENBAUM, 2002).

Por não se estabelecer uma comunicação cliente e servidor não é possível que usuários alterem ou compartilhem um mesmo arquivo ao mesmo tempo, como ocorre

por exemplo em um banco de dados (TANENBAUM, 2002). Enquanto que na conexão cliente e servidor, um nó é responsável por gerar recursos para os demais nós da rede (KUROSE JAMES F.; ROSS, 2017), isto é, em servidores dedicado um nó é especializado em um só tipo de tarefa, de modo que as requisições dos demais nós da rede são atendidas rapidamente. Tal estruturação apresentar explicita hierarquização do processo de comunicação, centralizando-se os processos de administração e configuração, e por conseguinte melhorando a organização e segurança da rede (HAYKIN, 2001). Normalmente são empregados servidores de comunicação (*backends*) entre as rede interna *LAN* e uma outra rede externa. Em tal estrutura se baseiam as conexões com *backbone* da internet por meio de provedores, referidos como *Internet Service Provider (ISP)* (MOTTA, 2012), permitindo-se estruturação hierárquicas de redes maiores (PERLMAN, 1999). Essa escalabilidade representa um papel fundamental no desenvolvimento de rede de computadores uma vez que com advento da globalização o sistema de redes foi concebido de modo a apoiar o crescimento a uma dimensão global gerando o próprio conceito de *internet* que se tornou uma tecnologia que molda a sociedade moderna (KIM, 2020).

Como explicitado, a estruturação de uma rede se dá por meio de nós e *link*, no entanto certos nós executam diferentes funções em uma determinada topologia de rede, existindo algoritmos de avaliação de robustez de uma rede pela importância de diferentes nós existentes de acordo com o parâmetros e topologia da rede (Zhou et al., 2018). Por exemplo, a Figura 1 mostra um conjunto de nós representado por computadores com cada um deles ligado por meio de uma ou mais ligações ponto a ponto. Dessa forma, os nós que realizam mais de uma conexão, isto é, compartilham pelo menos 2 *link* de comunicação distintos, devem organizar os dados recebidos e encaminha-los entre uma ligação para outra, processando pacotes de dados com baixa latência e limitados pela largura de banda disponível no canal (Fu, 2011). Esses nós responsáveis pelo reencaminhamento formam uma rede comutada ou *switched network*, destacada na Figura 1 (PETERSON, 2003). Além disso, a comutação em uma rede de computadores ajuda a determinar a melhor rota para a transmissão caso existam múltiplos caminhos ao longo da topologia de rede (WHITE, 2018). Atualmente as rede de comutação são formadas por uma quantidade elevada de nós, de modo que grandes redes comutadoras de dados não estão mais restritas ao meio corporativos fornecedores de serviços online e constituintes de grandes redes de centros de dados (MOTTA, 2012). Uma vez que ambiente de redes presentes em grandes universidades estão agregando um grande número de nós de computação e dispositivos de encaminhamento de dados, de modo a suportar os seus serviços internos como servidores *web*, *e-mail*, plataforma do sistema único integrado e bibliotecas online (Khattak et al., 2020).

Figura 1 – Topologia de uma rede comutada



Fonte: (PETERSON, 2003)

Existem diferentes tipos de redes comutadas, podendo ser classificadas quanto a comutação por circuito, por mensagem ou por pacote, sendo essa última categorizada em abordagem por data-gramas ou circuito virtual (WHITE, 2018).

Na técnica de comutação por circuito existe um caminho dedicado e estabelecido entre o nó que está enviando os dados e o nó que está recebendo, de modo que antes dos dados serem transferidos uma conexão é estabelecida, como ocorria por exemplo no sistema de telefonia fixa (TANENBAUM, 2002).

A comutação por mensagem se utiliza do mecanismo de armazenamento e encaminhamento de mensagem, isto é, a mensagem é transferida como uma unidade completa e encaminhada se utilizando de nós intermediários para o armazenamento interno na memória, processamento e encaminhamento (DUATO; YALAMANCHILI; NI, 2003). Ora, caso a mensagem seja maior do que a capacidade do canal, essa é fragmentada em agrupamentos menores, sendo cada agrupamento transmitido ao nó intermediário, responsabilizando-se por receber toda informação armazenar e reconstruir os dados para só então encaminhar a mensagem (PETERSON, 2003).

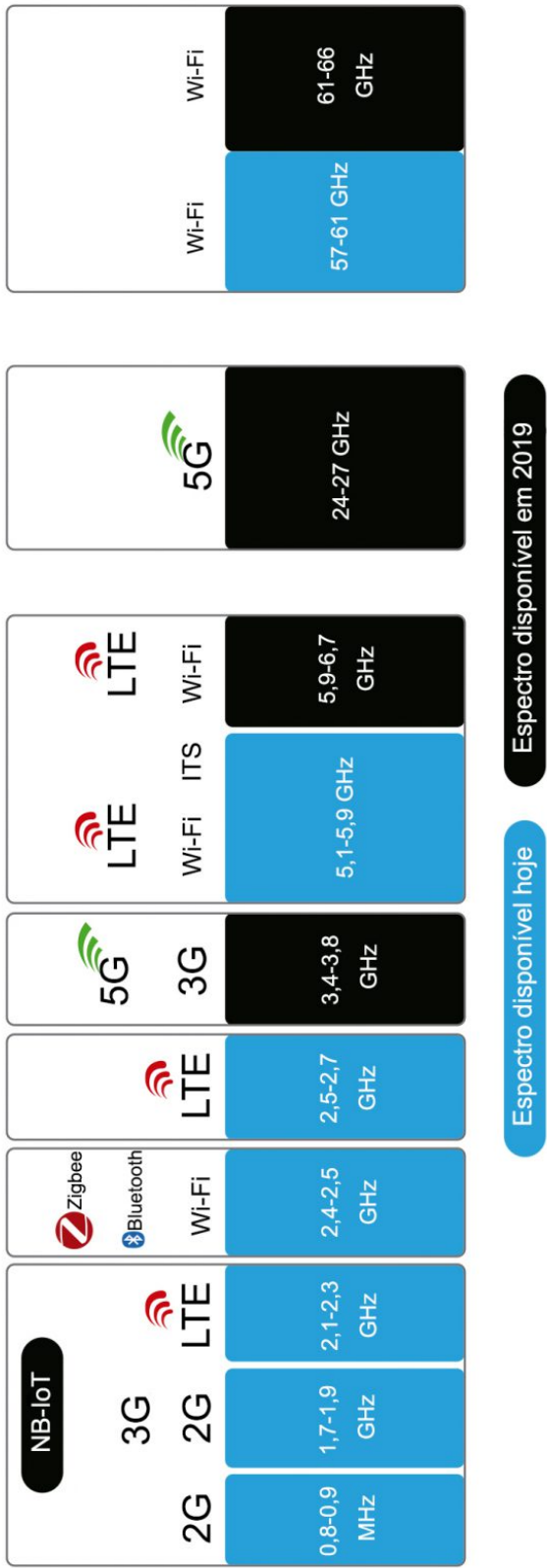
Dessa forma, tal aplicação não é viável em formas de comunicação em tempo real contemporâneas uma vez que é necessário aguardar que o nó intermediário colete todos os dados segmentados e os reconstrua antes de encaminhar a mensagem (PETERSON, 2003). Logo, implementou-se a técnica de comutação por pacotes, sendo essa a forma de

comutação adotada pela *internet*, da mesma forma, a mensagem é fragmentada em pacotes menores, no entanto cada pacote é enviado individualmente e possui informação sobre a sua origem e o *IP* de destino (TANENBAUM, 2002).

O tratamento de dados em redes de computadores lida na maioria das vezes com a nomenclatura de pacotes no âmbito digital, uma vez que os dados são gerados internamente e atuam em diferentes níveis da pilha de protocolo *TCP/IP*, no entanto para transmissão a informação é modulada e transmitida pelo canal (HAYKIN, 2001). Os sinais são ondas eletromagnéticas se propagando a velocidade da luz, de modo que a propagação de ondas eletromagnéticas mantém uma dependência com o meio, desse modo ao se propagar no caso das conexões *Point-to-point links* através do cabo de par trançado (cobre) ou fibra se obtém uma fração da velocidade da luz no vácuo (PETERSON, 2003). Diferentes tipos de redes são empregadas contemporaneamente aproveitando-se da característica eletromagnética de propagação (Amiri et al., 2012).

O aspecto eletromagnético possibilita a definição de bandas de comunicação normalmente por meio da frequência em *hertz* com o qual a onda oscila (Amiri et al., 2012). De modo que a crescente quantidade de espectro de rádio disponibilizado pelas autoridades reguladoras, desempenharam papel fundamental no crescimento do emprego de tecnologias sem fio (Zehl; Zubow; Wolisz, 2017). A Figura 2 mostra o espectro disponível atualmente bem como sua distribuições nos diferentes tipos de tecnologias empregadas.

Figura 2 – Bandas de frequência atualmente disponíveis e bandas de frequência atribuídas previstas para as principais tecnologias sem fios



Fonte: Adaptado de (5G..., 2020)

2.1.1.1 Wi-Fi

Contemporaneamente as conexões não estão exclusivamente restritas ao meio físico, de modo que o termo canal associado ao tipo de conexão pode ser associado ao espaço livre em uma conexão *Wi-Fi* (CHALLOO et al., 2012). O órgão de padronização para comunicação de dados de computador sem fio é o *IEEE* 802.11 (MALIK et al., 2015), tais padrões operam em frequências não licenciadas especificadas para aplicações *Industrial, Scientific and Medical (ISM)* (MEHDIZADEH, 2015), como mostrado no espectro da Figura 2, com a tecnologia *Wi-Fi* sendo empregada em 2.4GHz e 5GHz.

Tal característica permite na teoria de redes se definirem diferentes bandas de operação dos diferentes tipos de dispositivos (D.E., 2000). De modo que ao longo do desenvolvimento tecnológico do *Wi-Fi* foram desenvolvidas várias versões do padrão de comunicação de dados sem fio 802.11 (MALIK et al., 2015). Inicialmente o *Institute of Electrical and Electronics Engineers (IEEE)* anunciou o *IEEE* 802.11 como sendo a norma *LAN* sem fios em 1997, devido a sua flexibilidade este tem sido constante atualizado e otimizado desde então para se ajustar às necessidades contemporâneas, originando diferentes padrões *LAN* e *WLAN* como 802.11n, 802.11a, 802.11b, 802.11c (Hu; Zhu; Xiao, 2013).

Os padrões *WLAN* definidos pela *IEEE* pertencentes à família *IEEE* 802.11 são normalmente conhecidos pela sua denominação comercial *Wi-Fi* sendo especificado quanto a máxima velocidade de transmissão, a frequência de operação e a técnica de modulação empregada (CHALLOO et al., 2012). A modulação é a técnica responsável por adequar o sinal transmitido a frequência a ser empregada na transmissão, diferentes técnicas resultam em diferente eficiência espectral, isto é, a relação da taxa de transmissão em *bits* por segundo com a frequência de transmissão em Hertz (HAYKIN, 2001).

2.1.1.2 802.11n

A norma recente 802.11n amplamente utilizada atualmente apresenta modulação *Orthogonal Frequency-Division Multiplexing (OFDM)* na frequência de 5GHz que em vez de transmitir os símbolos de forma ordenada em um canal, essa técnica de modulação fragmenta a largura de banda do sistema em vários sub canais, chamados de sub portadoras, transmitindo os símbolos de forma simultânea (HAYKIN, 2001). Tal técnica é possível devido à ortogonalidade, isto é as subportadoras devem apresentar um número inteiro de ciclos dentro do símbolo *OFDM* de modo que não ocorra interferência entre subportadoras adjacentes *Inter-symbol interference (ISI)* (GROSS et al., 2009). A norma 802.11n ainda emprega tecnologias, como *Multiple Input Multiple Output (MIMO)* operando com múltiplas entradas e múltiplas saídas, para atribuição de maior número de canais disponíveis e maior largura de banda de modo a melhorar significativamente a taxa de alcance das redes locais sem fios (*WLANs*) (GONG; ZHAO; YANG, 2014).

A popularidade das redes *Wi-Fi* aumentou significativamente devido à sua capacidade de proporcionar uma forma eficiente e prática de ligação de uma multiplicidade de dispositivos (CHALLOO et al., 2012). Tal popularidade das redes sem fios é constantemente impulsionada pela presença de dispositivos portáteis presentes no cotidiano cuja conectividade é possibilitada pela conveniência da comunicações sem fios (MALIK et al., 2015). Assim, a escalabilidade da rede de *internet* muito se deve ao *Broadcast* (BALASUBRAMANIAN, 2020). Ora, o *Broadcast* permite o endereçamento de pacotes para todos os destinatários simultaneamente ao usar um código especial no campo de endereçamento, logo, quando um pacote codificado assim é transmitido este é recebido e processado por todos nós da rede (Nguyen et al., 2009).

2.1.2 Endereçamento

A estrutura da *internet* é hierárquica, e altamente complexa do tipo *backbone* (MOTTA, 2012). A *LAN* interna deve-se conectar à *backbone* de alguma forma de modo usufruir dos serviços da *internet*. A conexão se dá por meio de um servidor intermediário provedor de serviço, *Internet Service Provider ISP* sendo tal rede responsável pelo controle e gerenciamento dos endereços de *IP* atribuídos às suas subordinações (NUECHTERLEIN, 2005). Além disso, o protocolo *TCP/IP* é roteável, ou seja, foi projetado considerando a interligação de diversas redes (ALPERN; SHIMONSKI, 2010), uma vez que o conceito de roteamento infere diferentes caminhos entre o transmissor e o receptor (WHITE, 2018). Dessa forma, utiliza-se de uma implementação por meio de endereçamento lógico denominado endereçamento *IP*, isto é, difere-se de acordo com a lógica ou localização da *LAN*, podendo ser designado manualmente ou dinamicamente por meio de protocolos (INSAM, 2003). Dessa forma, elimina-se a dificuldade da heterogeneidade, isto é, a dificuldade de implementação que surgiria em estabelecer comunicação entre redes construídas de formas distintas com diferentes topologias (PETERSON, 2003). A pilha de protocolos *TCP/IP* é projetada de modo que seja possível identificar cada dispositivo conectado a uma rede por meio de um único endereço *IP* (KUROSE JAMES F.; ROSS, 2017).

2.1.2.1 Endereçamento *IPv4*

O endereço de *IPv4* é um número de 32 *bits*, representado de forma decimal em 4 campos de 8 *bits* cada, ou 1 *byte* (WHITE, 2018). Como cada campo possui 4 *bits*, tem-se que os valores que podem ser atribuídos a cada campo vão de 0 até 2^8 , ou seja 255. Logo, teoricamente o menor endereço de *IP* é 0.0.0.0 e o maior 255.255.255.255 (PETERSON, 2003). Além disso, os endereços de *IP* são estruturados de forma hierárquica, isto é, especificação do campo indica a rede, de modo que todos nós conectados a essa rede apresentem sintaxe semelhante até esse campo (D.E., 2000). Ora, a outra parte é indicativa ao *host*, isto é, computador ou dispositivo conectado a uma rede, de modo que

tal estruturação permite distinção de cada *host* dessa rede de forma unívoca (ALPERN; SHIMONSKI, 2010). Dessa forma, para facilitar a distribuição de endereços *IP* foram especificadas classes de endereços explicitadas na Tabela 1 (FARREL, 2004):

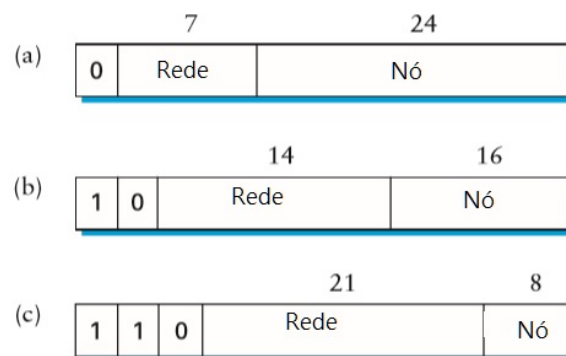
Tabela 1 – Faixa de endereçamento *IPv4*.

Classe	Menor endereço	Maior endereço
A	1.0.0.0	127.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.225
D	224.0.0.0	239.255.255.255
E	240.0.0.0	255.255.255.254

Fonte: (FARREL, 2004)

No endereçamento *IPv4* de classe A o primeiro *byte* é indicativo da rede, enquanto os outros três distinguem o *host*, dessa forma consegue-se endereçar 2^{24} dispositivos (TANENBAUM, 2002). Enquanto que para endereçamentos do tipo classe B são fixados os 2 primeiros campos e especificados para identificação da rede, restando 2 campos para atribuição de endereço á dispositivos (ALPERN; SHIMONSKI, 2010). Mostra-se na Figura 3 a disposição dos 32 *bits* disponíveis para endereçamento de acordo com a classe A,B ou C (PETERSON, 2003).

Figura 3 – Endereçamento *IPv4* de classe A (a), B (b) e C (c)



Fonte: Adaptado de (PETERSON, 2003)

Dessa forma, nota-se que a escolha da classe está intrinsecamente relacionada ao projeto, topologia e arquitetura da rede em questão (WHITE, 2018). A maioria das redes do tipo *LAN* atualmente se utilizam do endereçamento *IPv4* de classe C, de modo que se é possível endereçar até 254 dispositivos (os endereços 0 e 255 são reservados) (PETERSON, 2003).

Teoricamente seria possível estabelecer manualmente qualquer endereço de *IP* para os dispositivos em uma rede *WLAN* (WHITE, 2018), no entanto, para ter acesso á

internet a rede projetada deve se conectar ao *backbone* da rede mundial de computadores de alguma forma (MOTTA, 2012). Ora, como qualquer outra rede a *internet* é formada por comutadores como *switch*, *router* entre outros dispositivos categorizados como nós que se comunicam entre si, sendo essa conexão estabelecida chamada de *link* (PETERSON, 2003).

O tipo de conexão estabelecido é distinguido em duas categorias principais, uma referente ao transporte e outra ao acesso (WHITE, 2018). O *link* de transporte, é responsável pela ligação efetiva entre os nós de diferente rede suscetivamente, originando a chamada "espinha dorsal" da *internet*, justificando-se a referencia a essa classe de rede como *backbone* da *internet* (MOTTA, 2012). Além disso, as conexões também podem desempenhar funcionalidades de facilidades de acesso, são *links* responsáveis pela chamada conectividade de "última milha", isto é, a ligação responsável pela conexão do usuário final com um nó de rede e posteriormente com o *backbone* (NUECHTERLEIN, 2005). Essa conexão se dá por meio de provedores de *internet* *ISP*, fornecendo a ponte entre sua rede doméstica e os diversos servidores *backbone* estruturados na rede mundial de computadores (DANIEL; DANIEL, 2012).

A estrutura da *internet* é hierárquica e a quantidade de endereçamento de *IPv4* é limitada por 32 *bits* 2^{32} combinações que dependendo da classe empregada pode permitir diferentes distribuições (FARREL, 2004), de acordo com a Tabela 2. Por exemplo, com endereçamento *IPv4* de classe A é possível atribuir 128 redes (redes domésticas do tipo *LAN*) com 16 milhões de *host* cada, ou então para atribuições de classe B 16384 redes com até 64000 *host* em cada rede, e finalmente 2 milhões de redes com até 256 *host* possível em cada rede, para classe C (TANENBAUM, 2002).

A distribuição dos números de redes disponíveis é dado por uma corporação sem fins lucrativos, *Internet Corporation for Assigned Names and Numbers* (ICANN) de modo a se evitar redundância em um endereçamento (FARREL, 2004). Por sua vez, a *ICANN* distribuí diferentes espaçamento de endereços para diversas autoridades de diferentes âmbitos, tendo como destinatário os grandes provedores de *backbone* que vendem serviços de transporte aos *ISP* (NUECHTERLEIN, 2005). Estes provedores são responsáveis então pela atribuição de *IP* às suas camadas de redes subordinadas pela interface de facilidade de acesso (TANENBAUM, 2002), logo seria inevitável um conflito de *IP* caso uma rede doméstica fosse conectada diretamente á *backbone* da *internet* com um endereço de *IPv4* na *LAN* atribuído genericamente e sem padronização (PETERSON, 2003).

Surge então uma faixa de endereços conhecidas a priori (FARREL, 2004), as faixas são mostradas na Tabela 2 e definidas pela *RFC5735*¹ (NUECHTERLEIN, 2005).

Ora, os roteadores presente na rede de *internet* são configurados de modo a detectar

¹Disponível em: <<https://tools.ietf.org/html/rfc5735>>

esses endereços como indicadores de uma rede particular, referenciado normalmente como *localhost* (PERLMAN, 1999), barrando-se assim o encaminhamento de pacotes que referenciem essas faixa para a *internet* (WHITE, 2018).

Tabela 2 – Endereços especiais *IPv4* reservados para *LAN* .

Classe	Menor endereço	Maior endereço
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Fonte: (WHITE, 2018)

Assim, para se conectar uma rede *WLAN* como a do laboratório LPS a *internet*, baseia-se no endereçamento de classe C, uma vez que o número de *bits* referentes ao endereçamento de nós são suficientes para as especificidades desejadas (Zehl; Zubow; Wolisz, 2017). Dessa forma, somente o último *byte* do campo *IPv4* é utilizado para endereçamentos dos dispositivos conectados a essa rede (KUROSE JAMES F.; ROSS, 2017). Logo, os dispositivos são configurados com o endereço *IPv4* unívoco da forma: 192.168.1.x , onde x é um número natural entre 2 e 254 (WHITE, 2018).

A nível de protocolos de comunicação o primeiro *byte* no campo de endereçamento lógico já fornece informação ao roteador de que se trata de um endereço da Classe C e então deve ser tratado de modo interno á rede, como *localhost* (VASSEUR; PICKAVET; DEMEESTER, 2004). Ora, essa informação esta presente na estrutura de dados do pacote, ou *Protocol Data Unit (PDU)*, por meio do encapsulamento, referenciada como pacote (*packet*) na camada de rede pertencente á estruturação da pilha de protocolos *TCP/IP* (KUROSE JAMES F.; ROSS, 2017).

O *ISP* designa um *IP* fixo ao usuário, normalmente funcionando de forma dinâmica, isto é, podendo ser diferente a cada conexão (WHITE, 2018). É importante salientar que o *IP* utilizado na topologia interna da rede *WLAN* ou *LAN* não coincide com o endereço público fornecido pelo *ISP* (INSAM, 2003). Por exemplo, em uma rede *WLAN* de determinada topologia os *hosts* tem endereço *IP* atribuído da forma 192.168.y.x (com x, y números naturais expresso na base decimal e pertencente ao intervalo de 2 – 254), na Figura 4 é exemplificado por meio da ferramenta da *CISCO*, o *packet tracer*². Implementa-se uma topologia simulada funcional em que é explicitada a relação de cada campo do protocolo *IPv4* com a rede interna bem como quanto ao endereçamento de dispositivos, pelos campo atribuídos ao endereço *IPv4* de Classe C em uma rede doméstica.

enquanto que o *IP* fornecido pelo *ISP* pode adotar qualquer valor fora do trecho restrito aos endereçamentos especiais explicitados na Tabela 2. Nota-se que a classe referente a esse tipo de endereçamento atribuído pela *ISP* não é mais C. Ora, como o *ISP*

²Disponível em: <<https://www.netacad.com/courses/packet-tracer>>

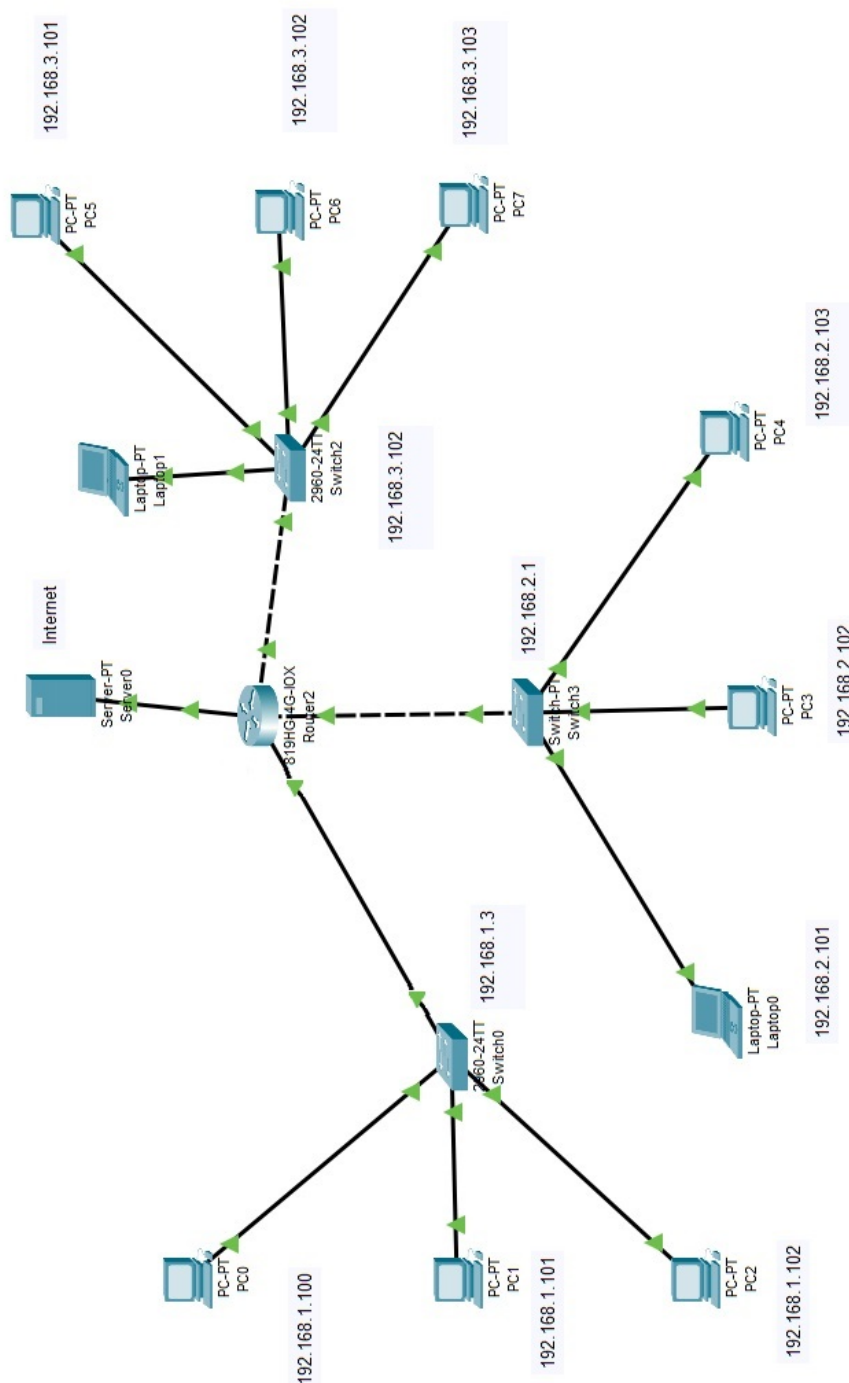
conecta uma rede doméstica ao *backbone* da *internet* sendo responsável pela atribuição e gerenciamento de pacotes de dados de diferentes topologias de redes é necessário uma maior flexibilidade na quantidade de redes disponíveis, uma vez que com endereçamento tipo C, exclui-se também todo o bloco de endereços atribuído a 127.0.0.0/8 na notação *Classless Inter-Domain Routing (CIDR)* como faixa de endereço de *loopback* pelo IETF (WHITE, 2018).

No entanto, quando é necessário especificar o *loopback*, o endereço *IPv4* mais utilizado é o 127.0.0.1 (VASSEUR; PICKAVET; DEMEESTER, 2004). Essa estrutura é o endereço de acesso próprio, utilizada para detecção de falhas na maioria dos sistemas (PETERSON, 2003). No entanto, em *linux* o endereço de *loopback*, designado por *lo0*, associa-se á uma interface virtual de rede, *Virtual Network Interface (VIF)*, sem necessariamente atribuir fisicamente a um dispositivo de rede (WHITE, 2018). Dessa forma, uma vez que uma aplicação como o SSH utilizar-se dessa interface passará a enviar ou receber tráfego usando o endereço *IP* designado pela interface virtual e gerado na sua atribuição, em vez do encaminhamento padrão pelo endereço de *IP* fornecido pelo *ISP*, prática comumente conhecida como tunelamento (KUROSE JAMES F.; ROSS, 2017). Nota-se que é por meio do *Dynamic Host Configuration Protocol (DHCP)* e a partir do *IP* do provedor que está associado ao *gateway* e por conseguinte ao *NIC* do computador que o tráfego originalmente se concentra (NEGUS, 2020). Por meio do protocolo *DHCP*, atribuído nas *RFC* 2131³ e 2132⁴ é possível que mais de um dispositivo em uma rede interna acesse a internet por meio do endereço de *IP* público fornecido pelo provedor *ISP* (Wang et al., 2009). Ora, o *DHCP* no roteador atua como um servidor especial que atribui endereços de *IP* a *host* solicitantes localizados na mesma rede local que o roteador (BALASUBRAMANIAN, 2020).

³Disponível em: <<https://tools.ietf.org/html/rfc2131>>

⁴Disponível em: <<https://tools.ietf.org/html/rfc2132>>

Figura 4 – Exemplificação de endereçamento *IPv4* de classe C (c)



Fonte: Elaborado pelo Autor

Com a popularização da internet e a estruturação globalizada contemporânea a quantidade de dispositivos conectados aumentou exponencialmente, bem como às aplicações que requerem vários dispositivos conectados simultaneamente por meio de diferentes endereços de *IP* (WHITE, 2018). Logo, percebeu-se que os 4 *bytes* de endereçamento de *IPv4* não seriam mais suficientes, como por exemplo o advento da internet das coisas

de *IoT* (YAN; LEE, 2019). Ora, como a aplicação de internet das coisas, *Internet Of Things (IoT)* consiste em uma rede de dispositivos físicos capazes de se comunicar interna e externamente, tais dispositivos apresentam diferentes capacidades computacionais e sensoriais, proporcionando interações complexas com a rede (CHAE, 2019). A aplicabilidade de tal tecnologia impulsionou um crescimento exponencial da necessidade de endereçamento à medida que o número de dispositivos aumenta vertiginosamente e deverá atingir vários milhares de milhões em 2020 (TOURNIER et al., 2020). Como o endereçamento por *IPv4* atingiu uma limitação quanto à tangibilidade do número de dispositivos globalmente conectados optou-se pela padronização do endereço de *IP* utilizando 128 *bits*, chamado *IPv6* (ALPERN; SHIMONSKI, 2010).

Além disso, a cada nó em uma rede *LAN* é atribuído outro tipo de endereçamento, por meio de *Media Access Control Address (MAC)*, sendo designado pelo fabricante em formato hexadecimal e composto por 48 *bits* (PERLMAN, 1999). Sendo esse único para cada fabricação de *Hardware* e estático, isto é, não sendo possível alterar um *MAC* em uma *LAN*, por isso sendo chamado de endereço físico (KUROSE JAMES F.; ROSS, 2017). Ora, o Endereçamento de *MAC* se dá por meio de *switch* e própria *NIC*, é atribuído a dispositivo que atuam nas últimas camadas da pilha de protocolo *TCP/IP* e desconhecem atribuições lógicas de camadas superiores como a camada de rede em que atua o protocolo *IP* (INSAM, 2003). Dessa forma, em uma *LAN* o *switch* entrega o pacote de dados para o destinatário baseado no endereço *MAC* do dispositivo físico. Ora, cada *switch* contém uma tabela de endereços *MAC* de modo a conhecer a qual destino o pacote de dados deve ser encaminhado (INSAM, 2003). Dessa forma, mesmo o roteador que endereça inicialmente os pacotes de dados á uma rede por meio de endereço de *IP* se utiliza em algum momento de *MAC* (BALASUBRAMANIAN, 2020). Ora, uma vez encaminhados os pacotes de dados são entregue subsequentemente aos roteadores até ao destinatário correto por meio do endereçamento físico dos dispositivo (*MAC*). Atualmente a maioria dos *switch* e o roteadores apresentam internamente o protocolo *UPnP*, ou *Universal Plug and Play* de modo a permitir que os dispositivos sejam automaticamente detectados na rede facilitando a configuração e gerenciamento (WHITE, 2018).

No entanto, o que vai garantir a entrega do pacote de dados á aplicação correta que está requerendo dados neste nó é o endereçamento por porta (TANENBAUM, 2002). Uma porta de rede é uma estrutura lógica que identifica e especifica um processo ou serviço de rede, sendo a origem ou o destino do processo de endereçamento, dessa forma está associado ás aplicações sendo executadas no cliente ou servidor. Uma vez que a cada aplicação é associado um número de porta específico e atua na camada de aplicação da pilha de protocolos *TCP/IP* (KUROSE JAMES F.; ROSS, 2017). No entanto, o conceito de porta está intrinsecamente associada ao protocolo de comunicação empregado *TCP/UDP* bem como ao endereço de *IP* do *host* especificado um número de porta (ALPERN; SHIMONSKI, 2010). Ora, ao criar uma aplicação é designado um tipo de *socket*, podendo ser este *TCP*

ou *UDP*, e como apresentam características e serviços diferentes são usados em diferentes implementações, por exemplo o protocolo *UDP* não fornece entrega confiável de pacotes ou controle de tráfego como *TCP* sendo restrito a multiplexação e demultiplexação para executar a entrega de pacotes a diferentes aplicações apenas com um controle de erro *checksum* presente no *Protocol Data Unit (PDU)* da camada de transporte (WHITE, 2018). Desse modo, portas com número menor do que 1024 são atribuídas a aplicações gerais e bem conhecidas, de forma que são utilizados números elevado em outras implementações de modo a evitar conflito, além disso podem ser filtradas utilizando o *firewall* (DEAL, 2004).

2.2 Linux

O *linux* é o sistema operacional de código aberto mais popular do mundo atualmente, com uma comunidade ativa e ampla aplicabilidade nas mais diversas áreas científicas. Opta-se por *linux* na construção do *Beowulf* principalmente por se tratar de código aberto, isto é, pode ser facilmente modificado, reordenado e ajustado para qualquer que seja a aplicação, além de apresentar estabilidade, escalabilidade e constante atualização (GROPP EWING LUSK, 2003).

2.2.1 Distribuições

Entre as distribuições de código aberto estão popularmente *Ubuntu*⁵, *Debian*⁶, *Fedora*⁷, *LinuxMint*⁸, *ElementaryOS*⁹, *Archlinux*¹⁰, *CentOS*¹¹, *OpenBSD*¹² (NEGUS, 2020). No ano de 2016, foi feito um levantamento com pelo *StackOverflow* com 56,033 programadores em 173 diferentes países de modo que desenvolvedores respondendo 45 perguntas sobre os diferentes tópicos da área mostrando uma porcentagem de utilização para distribuições em *desktop* de 56.7% *Ubuntu*, 8.8% *Debian*, 7.8% *LinuxMint*, 6.5% *Fedora* e 20.3% Outras distribuições. O levantamento completo abrange diferentes parâmetros tentando traçar o perfil dos utilizadores de *linux* (SYSTEM, 2020). Cita-se brevemente a disposição de algumas distribuições abaixo.

Fedora é uma distribuição lançada em 6 de novembro de 2003 e com ultima versão datada de 27 de outubro de 2020. Desenvolvida pela iniciativa independente *Fedora Project*¹³ cuja fundação se deu quando *Red Hat* desmembrou a *Red Hat Linux* em *RHEL*

⁵Disponível em <<https://ubuntu.com>>

⁶Disponível em <<https://www.debian.org>>

⁷Disponível em <<https://getfedora.org>>

⁸Disponível em <<https://linuxmint.com>>

⁹Disponível em <<https://elementary.io>>

¹⁰Disponível em <<https://http://www.archlinux.org>>

¹¹Disponível em <<http://www.centos.org>>

¹²Disponível em <<http://www.freebsd.org>>

¹³Disponível em <<https://docs.fedoraproject.org/en-US/docs>>

ou *Red Hat Enterprise Linux* (adquirida em 2019 pela IBM) (RED..., 2020a) e *Fedora* cujo foco consistirão em implementações de sistema operacional baseado em código aberto (WEB..., 2020a). Utiliza-se atualmente a distribuição *Fedora* em computadores pessoais, *desktop*, *notebooks* e sistemas de *clustering* por meio da implementação *Fedora Workstation* cujo ambiente de trabalho utilizado é GNOME¹⁴, o sistema operacional é focado em performance e estabilidade. Além disso, outras implementações são constantemente desenvolvidas como por exemplo, uma distribuição específica para servidores, inclusive com aplicabilidade em computação em nuvem por meio de *Fedora Servers*. Incluindo as últimas tecnologias empregadas em data center, como suporte a *softwares* populares como *node.js* e *go*. Outras distribuição também são empregadas em diferentes âmbitos, como por exemplo, utilizado-se em *dockers* não atualizáveis por meio de *Fedora Silverblue* ou ainda utilizado em aplicabilidade em internet das coisas ou *IoT* com *Fedora IoT*.¹⁵

Red Hat Enterprise Linux é a principal distribuição utilizada em empreendimentos corporativos. Apesar do código fonte ser fornecido *Red Hat* restringe o uso da sua distribuição por *trademark* (RED..., 2020b). Assim, surgem várias distribuições de terceiros baseadas no que tange às implementações gratuitas do *RHEL* como *CentOS*¹⁶ *Scientific Linux*¹⁷ e *Oracle Linux*¹⁸.

Manjaro é uma distribuição *linux* livre e de código aberto lançada em 10 de julho de 2011 com último lançamento da versão 20.1.1 *codename mikah* em 2 de outubro de 2020. Trata-se de uma distribuição baseada sistemas operacionais do tipo *Arch Linux* com foco principal em acessibilidade e simplificação. Ora, o sistema foca em usuários comuns e sistema é projeto de modo a apresentar funcionalidade imediata, isto é, sem maiores configurações diretamente a partir da instalação com software pré instalado de acordo com o conceito de "*straight out of the box*", isto é, uso imediato¹⁹.

2.2.2 Estrutura

O terminal é uma interface entre o sistema operacional e o usuário, a aplicação é iniciada quando o usuário realiza *login*, sendo a forma de usuários submeterem instruções ao computador, é responsável por interpretar os *scripts*, que são estruturas de códigos utilizados para criar novos comandos personalizados no *linux*, atribui-se a extensão *".sh"* para essa estrutura de arquivos (BARRETT DANIEL J.; SILVERMAN, 2009). A *Command Line Interface (CLI)* interface de linha de comando, caracteriza a potência e versatilidade do *linux*, é eficaz para o emprego de tarefas quando existe repetição ou uma grande quantidade de arquivos envolvidos, uma vez que é possível criar *scripts*,

¹⁴Disponível em <<https://www.gnome.org>>

¹⁵Disponível em <<https://getfedora.org>>

¹⁶Disponível em <<https://www.centos.org/>>

¹⁷Disponível em <<https://scientificlinux.org>>

¹⁸Disponível em <<https://www.oracle.com>>

¹⁹Disponível em <<https://manjaro.org/>>

utilizando-se de *wild cards* e *alias*, esses *scripts* podem ser posteriormente programados e implementados em múltiplas máquinas apresentando fácil gerenciamento através de acesso remoto (INFRASTRUCTURE... , 2020).

Uma grande quantidade de *scripts* estão disponíveis para utilização nos mais diversos *shell* (BARRETT DANIEL J.;SILVERMAN, 2009). Ora, o *linux* é código livre e aberto, de modo que qualquer um pode criar *scripts* promovendo alterações e melhorias no sistema de acordo com suas próprias necessidades, o que garante escalabilidade ao sistema gerando dessa forma grande variedade de implementações. O primeiro *shell* implementado foi o *Bourne(sh) shell* sendo esse o *shell* original do *UNIX*, servindo de base para várias outras futuras implementações, incluindo o *shell* mais popular, conhecido como *Bourne-Again Shell (BASH)* sendo este utilizado como padrão na maioria das distribuições *linux*, apresentando compatibilidade com os *scripts* em *Bourne Shell(SH)* (KIDWAI et al., 2020).

A capacidade de armazenar arquivos em uma unidade de disco é atribuída sob a nomenclatura sistema de arquivos, ou *File System (FS)* (MACH, 2020). Trata-se de um sistema hierárquico que permite a criação e gerenciamento de qualquer número de subdiretórios sob um diretório raiz identificado por "/" (Wang et al., 2009). Um sistema *linux* se utiliza do *Filesystem Hierarchy Standard (FHS)*, um padrão de hierarquia do sistema de arquivos (NEGUS, 2020). Ora, na instalação de uma aplicação é imperativo que os ficheiro sejam organizados de forma coerente e consistente para garantir o funcionamento, de modo que o sistema de arquivos fornece carácter preditivo aos desenvolvedores de *softwares* e usuários sobre a localização de ficheiros e diretórios de instalação (ABBOTT, 2013). O *FHS* especifica os diretórios bem como o seu conteúdo e é diretamente subordinado à raiz, sendo essa a estrutura inicial do diretório de arquivos de sistema do *linux* (MACH, 2020). Além disso, o *FHS* caracteriza os arquivos por meio de compartilháveis ou não compartilháveis. Ora, uma rede pode ser capaz de carregar arquivos por meio do *Network File System (NFS)* de modo que executáveis podem ser compartilhados por múltiplos usuários, no entanto algumas informações mantém exclusividades a um computador específico e não podem ser compartilhadas (ELTRINOS, 2020). Outras duas classes de arquivos apresentam separação entre estáticos e variáveis, sendo os últimos arquivos que são criados pelos usuários ou descarregados (BOTH, 2020).

Alem disso, ao diretório está associado um caminho, ou *pathname*, isto é, as diretivas de localização a partir da estruturação base do sistema de arquivo do sistema operacional para o diretório (BOTH, 2020). Dessa forma, um *pathname* é dito absoluto quando sua origem se dá no próprio diretório *root* do sistema, seguindo linearmente diretivas internas de subdivisões até o diretório de destino (Wang et al., 2009). Além disso um *pathname* pode ser classificado como relativo, isto é, quando é iniciado a partir do diretório atual, ou *working directory*, de modo que as sintaxes referente ao diretório atual estão associado com "." (MACH, 2020). Por exemplo, para acessar o diretório *bin* a partir do diretório *usr*

existem duas possibilidades, uma é passar como parâmetro para o comando `cd` o *pathname* absoluto daquele diretório com `cd /usr/bin`, outra forma a partir do *working directory*, ou diretório ativo, dessa forma a partir do diretório *usr*, utiliza-se `cd ./bin`. Nota-se ainda que `..` é utilizado como parâmetro para diretório anterior, assim `cd ..` retrocede um diretório tornando o diretório ativo (BLUM, 2020).

2.2.3 Comandos

Sumários indexado com os principais comandos utilizados em *linux* bem como suas aplicações e algumas exemplificações podem ser encontrados sob diferentes iniciativas online, com diversos níveis de detalhamento²⁰.

Entre os comandos notáveis vários fazem parte do núcleo de utilitários *GNU* ou *coreutils*²¹, tratam-se de conjunto de pacotes do projeto *GNU* contendo implementações para ferramentas básicas de um sistema operacional. Os comandos contidos no pacote do projeto GNU tem seu funcionamento detalhado em um projeto de decodificação do núcleo do utilitário²². Além disso, caso algum comando não esteja disponível nativamente na distribuição instalada pode-se obtê-lo por meio do repositório²³ com `yum install coreutils` em sistemas baseados em *RHEL* como *CentOS* e *Fedora*. Ou então, por `apt-get install coreutils` em sistemas baseados em *Deb* como *Ubuntu*, *Debian* e *Linux Mint* (COREUTILS..., 2020).

Comandos são seguidos por uma ou mais opções, que consiste de um simples carácter precedido por `-`. Tal opção modifica o comportamento do comando, ou adiciona diferentes opções por um ou mais argumento, isto é, o item em sobre o qual o comando age, refere-se a esse modificar como *flag* (ELTRINOS, 2020). É possível obter os modificadores de comando ao digitar `comando --help`, uma vez que além dos comandos provenientes do projeto GNU muitos outros suportam *flag* de opção longa, isto é, determinadas por palavras (BOTH, 2020). Entre a grande gama de comandos existentes em *linuxs* destacam-se alguns utilizados no desenvolvimento do projeto:

O comando `pwd`: mostra o diretório atual (*print working directory*) (MACH, 2020);

O comando `cd`: é utilizado de modo a alternar o diretório atual (*working directory*), para tal o comando utiliza a sintaxe `cd` seguido do caminho até o diretório de acesso, pode-se acessar diretamente um diretório subsequente dentro do diretório atual por `cd usr`, ou então seguir ramificação a ramificação a partir da árvore dos arquivos de sistema até o diretório desejado por meio de uma única linha, como por exemplo `cd /usr/bin` (BLUM, 2020). Alguns modificadores são mostrados abaixo:

²⁰Disponível em: <<https://linux.die.net/Linux-CLI/doc-index.html>>

²¹<<https://www.gnu.org>>

²²Disponível em: <<https://www.maizure.org>>

²³<<https://git.savannah.gnu.org/cgit/coreutils.gitcom>>

`cd`: utilizado para acessar e mudar de diretório;
`cd -`: retorna ao ultimo diretório selecionado;
`cd ..`: vai para o diretório anterior na arvore;
`cd ./pasta`: seleciona a pasta existente no diretório;
`cd ~`: muda diretamente para o diretório home do usuário;

O comando `ls`: mostra o conteúdo e atributos do diretório, lista o tipo de arquivo, mostra permissão, tamanho, data de criação etc (BLUM, 2020). Pode-se atribuir o *pathname* absoluto desejado e dessa forma o comando não atua exclusivamente no diretório atual (*working directory*), como por exemplo com `ls /usr/bin` (MACH, 2020). Algumas *flags* para o comando `ls` são mostradas abaixo:

- a* : lista todos arquivos, incluindo arquivos ocultos;
- A* : forma análoga -*a* sem listar diretórios;
- d* : junto com -*l* mostra detalhes sobre diretório em vez de seu conteúdo;
- F* : Indicador, mostra "/" se o nome eh um diretório;
- h* : referente á *human readable* lista tamanhos em ordem de grandezas K,MB,G;
- l* : mostra o conteúdo e atributos do diretório, lista o tipo de arquivo, mostra permissão, tamanho, data de criação etc;
- r* : mostra os resultados em ordem reversa;
- S* : mostra o resultado ordenado pelo tamanho;
- t* : mostra o resultado ordenado pela data de modificação;

Além disso, como saída no terminal diferentes cores indicam o tipo de arquivo associado. Por exemplo, diretórios são exibidos em azul, arquivos comprimidos são exibidos em vermelho, texto em branco, imagem em rosa, *link* em *cian*, amarelo para dispositivos, verde para executáveis e vermelho piscando indicam links quebrados (MACH, 2020).

O comando `file`: mostra uma breve descrição da extensão do arquivo. Nota-se que em *linux* a extensão do arquivo não necessariamente reflete seu conteúdo como em outros sistemas operacionais, dessa forma o comando `file` produz como saída no terminal uma descrição sobre o conteúdo do arquivo (BLUM, 2020). O comando `less` que permite visualização direta de arquivos. de texto (formato de arquivos que contem informações do sistema, scripts etc.) (BLUM, 2020);

O comando `cp`: é utilizado para copiar arquivos e diretórios. O argumento de `cp` arquivo *pathname* é capaz de copiar múltiplos itens (arquivos ou diretórios) para um diretório (BLUM, 2020). Algumas *flags* comuns são:

- a*: copia arquivos diretórios e todos seus atributos;
- i*: se essa opções esta especificada ira requerer permissão antes de sob reescrever um arquivo existente;
- r*: copia recursivamente diretórios e seus conteúdos, tal opções ou a opção -*a* são necessárias

quando copiando diretórios;

-u: quando copiando arquivos de um diretório a outro apenas se copia os arquivos que não existem ou são mais novos do que os arquivos existentes no diretório de destino;

O comando mv: mover ou renomear diretórios. Para mover um arquivo é utilizada a sintaxe mv filename dir sendo *dir* o *pathname* absoluto daquele diretório, para renomear um arquivo basta incluir o nome do arquivo seguido do novo nome mv filename new_filename (MACH, 2020).

O criador de diretórios mkdir: utilizado para criar diretórios, o argumento de mkdir nome, pode ser repetido como por exemplo mkdir dir1 dir2 dir3 (MACH, 2020). E rm para remover arquivos e diretórios (MACH, 2020);

O Comando cat: A nomenclatura é derivada do termo em inglês *concatenate*, traduzido como concatenar, uma vez que permite que arquivos sejam concatenados ou unidos, criados ou exibidos. A sintaxe é da forma cat Opcao arquivo, de modo que os modificadores de opção podem ser visualizados diretamente no terminal com a entrada cat --help (SHOTTS, 2019). O comando é mais comumente utilizado quando é necessário visualizar o conteúdo de um determinado arquivo, isto é, o conteúdo do arquivo que é passado como parâmetro para o comando é exibido como saída no terminal (MACH, 2020). O Comando nano e vim são editores de textos difundidamente utilizados.

2.2.4 Permissões

Trata-se de uma classe de usuário especial que permite promover qualquer alteração a um serviço ou programa em *linux*, bem como acessar arquivos com qualquer tipo de permissão, normalmente é atribuído a nomenclatura de super-usuário ou *root*, dependendo da configuração do sistema operacional a classe de permissão pode ser elevada por meio do prefixo -su ou ainda ser atribuído localmente na execução de um comando específico precedendo o comando por *sudo* (MACH, 2020).

Ao se utilizar do comando ls -l para exibição de saída em formato longo é mostrado um campo composto por 10 caracteres precedendo a exibição do nome do arquivo, de tal forma que esses caracteres são referidos como atributos do arquivo (BLUM, 2020). O primeiro carácter da esquerda para direita é o identificador de tipo de arquivo. Por exemplo, caso o campo de caracteres for iniciado com "d" trata-se de um diretório, iniciado com - "é um arquivo comum, "c" um arquivo de carácter especial etc (MACH, 2020).

Os nove campos subsequentes são divididos em 3 grupos, relativo às permissões associadas ao dono do arquivo, permissão de grupo e permissão geral (NEGUS, 2020). Dessa forma, os termos subsequentes são chamados de modo de arquivo, estão associados com operações de escrita, leitura e com permissão de execução do arquivo (BLUM, 2020).

Ora, "r" está associado com a permissão de leitura e quando associado á diretórios permite listar o conteúdo. Além disso, o carácter "w" está relacionado com a permissão de escrita, isto é, modificação do arquivo, quando associado á diretórios permite a modificação deste por meio de criação, exclusão, além de modificação de nomes de arquivos do diretório (SHOTTS, 2019). Enquanto "x" indica que o arquivo deve ser tratado como um programa e executado. Logo, uma saída do tipo `-rwxr-xr-x` indicaria que se trata de um arquivo comum, com permissão de escrita, leitura e execução pelo responsável pela criação do arquivo, além disso o arquivo pode ser lido e executado por todos os outros (MACH, 2020).

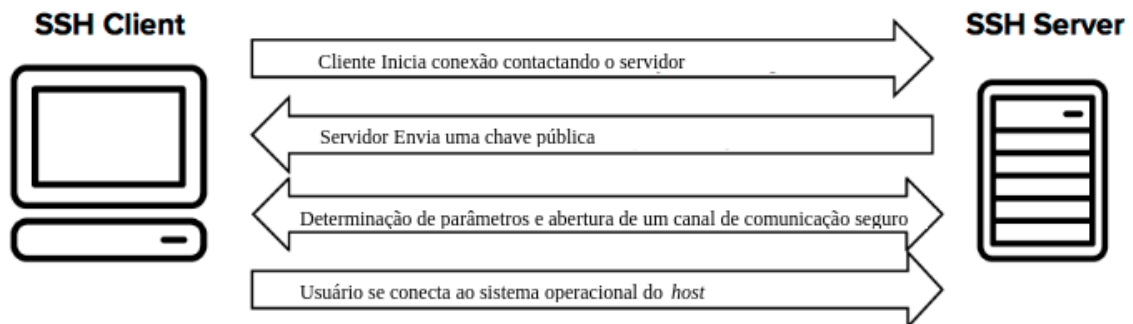
O comando `chmod` é utilizado de modo a se alterar a permissividade associada a um arquivo ou diretório (Wang et al., 2009). Dessa forma, apenas o responsável pela criação do arquivo ou um super-usuário pode atribuir permissões (SHOTTS, 2019). Como cada dígito em notação octal representa três números binários é utilizada a notação para atribuir diretamente 3 caracteres, uma vez que tem-se 3 variáveis associadas, isto é `rwx`, sendo variáveis booleanas, assim a notação octal permite a vantagem de associar um único atributo sem alterar os demais (MACH, 2020). Ora, ao construir a tabela verdade para 8 entradas tem-se todas combinações de 0 e 1 entre as variáveis e assim todas configurações possíveis entre os parâmetros de leitura, escrita e tipo de arquivo `7(rwx)`, `6(rw-)`, `5(r-x)`, `4(r-)` até `0(—)` (M., 2005). Por exemplo, a entrada 110 nessa tabela verdade $r=1$, $w=1$, $x=0$ representa 6 em decimal, de modo que é passado o valor decimal como parâmetro para o comando `chmod`, no entanto é necessário especificar os 3 grupos do modo de arquivo, sendo atribuído a cada um valor de 0 a 7, na ordem especificada nessa sessão (NEGUS, 2020). Isto é, `chmod 600 arquivo.txt` indica que 6 (110) foi atribuído ao grupo criador do arquivo, 000 como permissão de grupo e 000 ao grupo permissão global. Dessa forma, apenas ao criador do arquivo é permitido escrita e leitura não sendo esse arquivo um programa ou executável (BLUM, 2020).

2.2.5 SSH

SSH é um protocolo de rede criptográfico, de modo que permite operação de serviços e transferência de arquivos por meio de uma rede, na arquitetura cliente servidor (YLONEN, 2020b). Em termos de protocolos de internet o cliente é definido como quem inicia conexões e requisições com outros computadores enquanto o servidor é executado constantemente no *backend* da topologia aguardando requisições de conexão (MACH, 2020). Dessa forma a atuação do *SSH* está dividido em duas frentes, um servidor *SSH* sendo executado no *host* remoto esperando por conexões, por padrão na porta *TCP* 22, enquanto um cliente *SSH* é usado no sistema local para se comunicar com o servidor remoto (SHOTTS, 2019). Adicionalmente, toda comunicação entre cliente e servidor é encriptada, de modo que tentativas de identificação de qualquer um dos lados separadamente são ineficientes (INFRASTRUCTURE..., 2020). Ora, o pacote de informação trocado é

encriptado com chaves conhecidas apenas pelo cliente e servidor (CHAPPLE, 2020). A estruturação básica é mostrado na Figura 5, de forma que o serviço é caracterizado por atuar de forma segura por meio de uma rede não segura como a *internet*, sendo assim utilizado pela maioria dos *data center* ou grandes companhias comerciais (YLONEN, 2020e).

Figura 5 – Etapas de conexão *SSH*



Fonte: Traduzido de (YLONEN, 2020e)

Devido á questões de segurança quando se refere ao *SSH* na sintaxe do sistema operacional é referenciado a versão utilizada atualmente, o *SSH-2*, trata-se de uma releitura completa do protocolo original incorporando diversas medidas de proteção para vulnerabilidades apresentadas pelo *SSH-1* (CHAPPLE, 2020). Enquanto o protocolo *SSH-1* é monolítico, isto é, comprimido de modo a acumular múltiplas funções em um único protocolo, o *SSH-2* é separado em módulos e consiste de três protocolos atuando simultaneamente e definidos no *RFC* por: *SSH Transport Layer Protocol (SSH-TRANS)*²⁴, *SSH Authentication Protocol (SSH-AUTH)*²⁵ e *SSH Connection Protocol (SSH-CONN)*²⁶ (CHAPPLE, 2020). Ora, *RFC* é uma publicação da *Internet Society (ISOC)*²⁷ e está associada predominantemente ao *Internet Engineering Task Force (IETF)*²⁸ uma *Standards Developing Organization (SDO)* sendo assim a principal diretriz no desenvolvimento de padronização de protocolos para internet (YLONEN, 2020b).

Por meio disso, o protocolo *SSH-2* encripta todos os dados protegendo a conexão de interferência de terceiros, comumente referida na literatura como *man-in-the-middle attack*, uma vez que é necessário uma chave privada para realizar a autenticação, eliminando ainda ataques baseados em falsificação de *IP* e *Domain Name Service (DNS)* (YLONEN, 2020e). Entre, as técnicas de encriptação utilizada por *public-key* estão *Digital Signature*

²⁴Disponível em <<https://tools.ietf.org/html/rfc4253>>

²⁵Disponível em <<https://tools.ietf.org/html/rfc4252>>

²⁶Disponível em <<https://tools.ietf.org/html/rfc4254>>

²⁷Disponível em <<https://www.internetsociety.org>>

²⁸Disponível em <<https://ietf.org/>>

Algorithm (DSA), *Rivest-Shamir-Adleman (RSA)* e *Open Pretty Good Privacy (OpenPGP)* (CHAPPLE, 2020).

A maioria das distribuições utilizam a implementação *OpenSSH*²⁹ sendo utilitários de comunicação baseados inicialmente no protocolo *SSH*, o *OpenSSH* ainda suporta *SSH-2* tendo desvincilhado da sua base de códigos suporte ao *SSH-1* a partir da versão 7.6 (YLONEN, 2020e). A ferramenta é atualmente desenvolvida como parte do projeto *OpenBSD*³⁰, que consiste no desenvolvimento de um sistema operacional de código aberto baseado em *UNIX* e focado em segurança com criptografia integrada e portabilidade (YLONEN, 2020b). Dessa forma, ao se referir a cliente *SSH*, servidor *SSH* estamos nos referindo ao serviço ou *daemon* sendo executado com a implementação do *OpenSSH* utilizado atualmente e estruturado sob o *SSH-2* (CHAPPLE, 2020).

Dessa forma, para instalar o *OpenSSH* em uma distribuição *linux* é necessário instalar o pacote associado ao serviço, por meio da estrutura de pacote do sistema operacional explicitado na Sessão 2.2.8, com o comando `sudo dnf install openssh-server` é instalado o pacote *OpenSSH* e o *daemon* para o servidor (CHAPPLE, 2020). Além disso, como é característica de um servidor estar sempre disponível para requisições é interessante habilitar a inicialização da ferramenta *daemon* com o *boot*, ou inicialização do sistema operacional, por meio de `sudo systemctl enable sshd` (YLONEN, 2020b). Nota-se que *OpenSSH* requer a instalação do pacotes de dependência *Secure Sockets Layer (SSL)*, responsável por implementar bibliotecas importantes de criptografia, sendo instalado diretamente do repositório *fedora* por: `sudo dnf install openssl-libs`³¹ (INFRASTRUCTURE..., 2020).

2.2.5.1 Utilitários instalados com *OpenSSH*

Juntamente com o *OpenSSH* são instalados alguns utilitários de modo a facilitar a implementação e gerenciamento do servidor *SSH*. No entanto, ainda é necessário no cliente instalar um pacote adicional por meio de `sudo dnf install openssh-client`, implementando o *daemon* no cliente (INFRASTRUCTURE..., 2020).

Além disso, a ferramenta *ssh-keygen* é usada no cliente para criação de um par de chaves autenticadoras, usualmente referida na literatura como chaves *SSH*, sendo usado para *login* automático e autenticação do servidor (MACH, 2020). Ora, o protocolo *SSH* usa o método de criptografia com chave pública para autenticação servidor e usuário e essas chaves são geradas pelo comando `ssh-keygen -t algoritmo -b bits`, com os parâmetros e possíveis modificadores disponibilizados pelo próprio manual do comando³², sendo a opção `-t` utilizada para explicitar o algoritmo usado e `-b` a opção para escolher o número

²⁹Disponível em <www.openssh.org>

³⁰Disponível em <<https://www.openbsd.org>>

³¹Disponível em <https://fedora.pkgs.org/31/fedora-x86_64/openssl-libs-1.1.1d-2.fc31.i686.rpm.html>

³²Disponível em <<https://man.openbsd.org/ssh-keygen>>

de *bits* utilizado (YLONEN, 2020c). Entre os principais algoritmos utilizados encontra-se o *RSA*, um algoritmo antigo baseado na fatorização de números grandes, sendo o mais popular em conexões *SSH* (MACH, 2020), o uso para geração de assinatura digital e verificação foi especificado no *IETF RFC 8017*, sob o tópico *Public Key Cryptography Standard (PKCS)* (MORIARTY K.; RUSCH, 2020). Além disso, mantém um site oficial sobre implementações³³. Outro algoritmo possível de ser utilizado é o *Digital Signature Algorithm (DSA)*, trata-se de um antigo algoritmo de assinatura de digital governamental sob patente³⁴ e adotado como padrão federal sob número 186-4, *Federal Information Processing Standard (FIPS)*, o algoritmo se baseia na dificuldade de computar logaritmos discretos, normalmente são utilizado 1024 *bits* com verificação especificada no *IETF RFC 6979*³⁵ (PKCS..., 2020). O algoritmo mais recente entre os mais utilizados é o *Elliptic Curve Digital Signature Algorithm (ECDSA)*, padronizado sob *FIPS 186-5* com verificação especificada no *IETF RFC 64754*³⁶ é um novo padrão de assinatura digital padronizado pelo governo americano utilizando-se de curvas elípticas, dessa forma para gerar um par de *keygen* nesse algoritmo com *OpenSSH* utilizado-se a sintaxe `ssh-keygen -t ecdsa -b 521`, sendo `-t` indicador do algoritmo e `-b` especificando o numero de *bits* (INFRASTRUCTURE..., 2020).

O comando *Secure Copy (SCP)* é utilizado para copiar arquivo entre diferentes computadores utilizando o protocolo *SSH* sob a sintaxe `scp file host:path` (NEGUS, 2020). A implementação do *Secure Copy (SCP)* está relacionada com a disponibilidade do comando `glob`³⁷ (MACH, 2020), uma vez que a função `glob()` gera *pathnames* que implementam regras de coincidência para padrões na nomenclatura de arquivos usado pelo *shell* (CHAPPLE, 2020). Dessa forma, o arquivo especificado é copiado a partir do *host* (local ou remoto) e colocado no diretório indicado pelo *path*, caso esse não seja especificado o arquivo é copiado no diretório atual *working directory* (CHAPPLE, 2020). Além disso, as opções de origem e destino podem ser um *pathname* local, um outro *host* remoto com especificação opcional de diretório da forma `user@host:path` ou até mesmo uma *Uniform Resource Locator (URL)* na forma `scp://user@host:port/path` (SHOTTS, 2019). Demais opções disponíveis durante a cópia de arquivos estão no próprio manual do comando³⁸.

No entanto, a própria implementação *SSH* implementa a cópia de chaves por meio do utilitário *ssh-copy-id*, de modo que é copiada a chave pública do cliente para o servidor diretamente pela sintaxe: `ssh-copy-id -i ~/.ssh/id_rsa -p port user@hostname`, com os modificadores e parâmetros do comando disponibilizados pelo próprio manual do

³³Disponível em <<https://www.rsa.com/>>

³⁴Disponível em <<https://patents.google.com/patent/US5231668>>

³⁵Disponível em <<https://tools.ietf.org/html/rfc6979>>

³⁶Disponível em <<https://tools.ietf.org/html/rfc4754>>

³⁷Disponível em <[globhttps://man.openbsd.org/glob.3](https://man.openbsd.org/glob.3)>

³⁸Disponível em <<https://man.openbsd.org/scp.1>>

comando³⁹ (SHOTTS, 2019). Dessa forma, o comando é utilizado para se conectar ao servidor e efetuar o *upload* da chave pública do usuário no servidor, o comando edita o arquivo *authorized_keys* no servidor (NEGUS, 2020).

O *SSH File Transfer Protocol (SFTP)* é um protocolo de transferência de arquivo segura, é executado sob o protocolo *SSH* e suporta todas suas funcionalidades de segurança e autenticação (YLONEN, 2020b). Substitui o *File Transfer Protocol (FTP)* como protocolo padrão de transferência de arquivo uma vez que fornece todas funcionalidades desse de forma mais simples, segura e confiável (NEGUS, 2020). O comando é usado sob a sintaxe `scp -r file user@host:path` ou dependendo do fluxo de interesse da transferência `scp -r user@host:file path` (SHOTTS, 2019). Dessa forma, é copiado um ou mais arquivos para a conta do *host* especificado, se não for especificado o usuário então é assumido o mesmo nome de usuário por parte do cliente, dessa forma, os arquivos são copiados ao diretório especificado pelo *path* ou diretório *home* do usuário caso não seja especificado (INFRASTRUCTURE..., 2020). Ainda, é possível copiar-se recursivamente diretórios e todos subdiretórios por meio da flag `-r`. Demais opções disponíveis durante a transferência de arquivos estão no próprio manual do comando⁴⁰ (MACH, 2020). Além disso, *SFTP* pode ser utilizado de forma similar ao compartilhamento de arquivos nos *Windows* e *NFS* no *Linux*, com o diferencial de não ser restrito ao uso na rede local por ser encriptado e segura sob *Network address translation (NAT)* (MACH, 2020). Baseado nisso em *linux* é implementada a estrutura de arquivos de rede *SSH Filesystem (SSHFS)* cuja execução se dá sob o protocolo *STFP* (MACH, 2020), de modo que pode utilizar qualquer servidor *SSH* se utilizando de arquivos remotos sob a rede como se fossem gerenciados como arquivos locais, isto é, podem ser montados ou desmontados conforme o interesse sem configuração adicional pelo administrador do servidor (CHAPPLE, 2020).

2.2.5.2 Configuração *OpenSSH*

Além das ferramentas citadas na sessão anterior, com a instalação do *OpenSSH* é incluso nas configurações do sistema operacional o próprio serviço `/etc/sysconfig/sshd`, trata-se do processo do servidor *OpenSSH*, referenciado como *OpenSSH Daemon (SSHD)*, e é responsável por autenticar, encriptar, estabelecer conexões com terminal, transferir arquivo e rotear (INFRASTRUCTURE..., 2020).

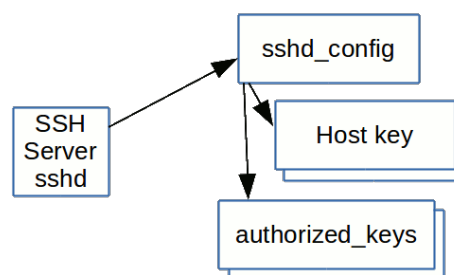
Quanto á configuração, existem dois conjuntos de arquivos de configuração, um especificado para o cliente com *SSH*, *SCP* e *STFP* e os especificados para o servidor com a aplicação *daemon* (INFRASTRUCTURE..., 2020). Dessa forma, as configurações de parâmetros de interesse do *daemon* são armazenadas sob diretório `/etc/ssh/` de modo que nesse diretório existe um arquivo de configuração da aplicação com nome *sshd_config*

³⁹Disponível em <<https://www.freebsd.org/cgi/man.cgi?query=ssh-copy-id>>

⁴⁰Disponível em <<https://man.openbsd.org/scp.1>>

(CHAPPLE, 2020). Esse arquivo de configuração especifica o serviço com opções de criptografia e autenticação, localização de arquivos, modos de *login* entre outros parâmetros (MULTIPLE... , 2020), todos os parâmetros associados podem ser encontrados no manual do comando⁴¹. De modo geral, o servidor executa a leitura de vários parâmetros ligados àquele *daemon* pelo arquivo de configuração, o *sshd_config*, que ainda especifica a localização de uma ou mais chaves de *host* bem como especifica a localização dos arquivos de chaves autorizadas para acesso de usuários naquele servidor, como o esquemático mostrado na Figura 6 (YLONEN, 2020e).

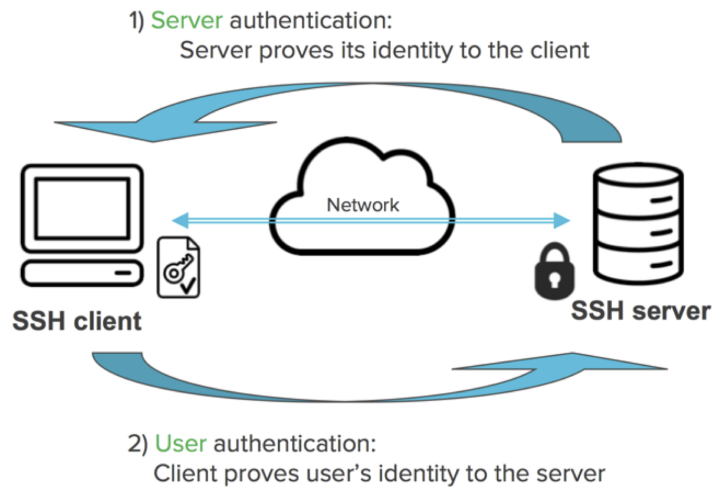
Figura 6 – Especificações do *sshd_config*



Fonte: (YLONEN, 2020e)

O *OpenSSH* fornece mecanismo de autorização baseado em chaves criptográficas individuais para cada par cliente servidor como mostrado na Figura 6 (YLONEN, 2020e). A autenticação por chave pública se mostra uma alternativa mais segura uma vez que não é necessário encaminhar efetivamente uma senha ao servidor, evitando assim a possibilidade de interceptação (YLONEN, 2020d), uma vez que a capacidade preditiva de identificação, referida na literatura como *Identity and Access Management (IAM)* é a fundação da estrutura de segurança da informação (YLONEN, 2020b). A criptografia por chave pública quando implementado por uma prática com os algoritmos de encriptação descritos interiormente tornam a conexão robusta e segura (INFRASTRUCTURE... , 2020). Uma vez que contemporaneamente as necessidades básicas de gerenciamento de acesso se dão quanto à identificação confiável de usuário, além da capacidade de controlar e gerenciar os recursos que serão acessados pelos usuários individualmente sob essas diretivas (CHAPPLE, 2020).

⁴¹Disponível em <http://man.openbsd.org/sshd_config>

Figura 7 – Autenticação cliente-servidor na conexão *SSH*

Fonte: (CHAPPLE, 2020)

O processo de gerenciamento de acesso do *SSH* se dá pela criação de um par de chaves na máquina do cliente com o comando `ssh-keygen -t ecdsa -b 521` que gera uma chave pública `id_ecdsa.pub` e uma chave privada `id_ecdsa` sob o diretório `/home/USER/.ssh/`, de modo que cada par é específico e unívoco para cada usuário (YLONEN, 2020d). Dessa forma, a chave privada é mantida na máquina do cliente, servindo de prova para identificação do usuário, enquanto a sua chave correspondente pública será alocada na máquina do servidor (NEGUS, 2020). Além disso, o comando permite atribuir nomes diferentes à chave na sua criação, por padrão é utilizado `id_ecdsa` nesse caso, é oferecida também a opção de atribuir uma senha pessoal no instante da geração, associando-a a esse par unívoco (INFRASTRUCTURE..., 2020).

O envio da chave pública do cliente para o servidor pode se dar por diferentes meios, como pelo utilitário *SCP* de transferência de arquivos descrito na sessão de ferramentas implementadas juntamente ao *OpenSSH* (CHAPPLE, 2020). Ou ainda pode ser realizado por *SCP*, além de incluídas manualmente diretamente no diretório do servidor (MACH, 2020). No entanto, do ponto de vista de gerenciamento é mais interessante a implementação por `ssh-copy-id -i ~/.ssh/id_ecdsa.pub user@hostname` uma vez que o comando já conecta ao servidor e efetua o *upload* da chave pública do cliente no próprio arquivo de chaves permitidas ao servidor, concatenando-o ao final do arquivo caso o mesmo já exista. Uma vez que o comando edita diretamente o arquivo `.ssh/authorized_keys` no servidor (NEGUS, 2020).

Tal implementação torna-se mais robusta quando utilizada em conjunto com atribuições de *wild card* de modo que gerenciam-se um maior quantidade de chaves (INFRASTRUCTURE..., 2020). Ora, ao se especificar as chaves por `ssh-copy-id -i ~/.ssh/id*.pub user@hostname` (substituindo parcialmente a especificação de nome da chave por *)

o comando irá copiar todos arquivos iniciados por *id* com extensão ".pub". A utilização justifica-se devido á própria sintaxe de nomenclatura atribuída na criação das chaves com *ssh-keygen* associada com a sintaxe de implementação do código do *ssh-copy-id*. Garante-se assim que apenas as recentes modificações de chaves pública criados no cliente e ainda não instalada no servidor serão adicionadas ao arquivo de autorização sob qualquer algoritmo de encriptação (Wang et al., 2009).

A partir dessa etapa apenas um usuário possuindo uma chave privada que mantém correspondência com uma chave pública armazenada no computador do servidor e autorizado pela senha pessoal atribuída no instante da criação da chaves criptográficas estará apto a se autenticar no servidor (YLONEN, 2020b).

Por padrão as permissões associadas aos arquivos de chaves são `rwX-----` ou `700` expresso na notação octal como explicitado na Sessão 2.2.4, quando abordado permissões com comando `chmod`. Ora, devido ao tipo de implementação de servidor *cluster* é interessante não permitir conexão como *root* uma vez que com `chmod 700` apenas o criador do arquivo seja capaz de manipular a visualizar ou alterar a chave (YLONEN, 2020a).

Além disso as opções de configuração estão sob o arquivo de configuração do *daemon*, `/etc/ssh/sshd_config`. Tratam-se das configurações do servidor, e entre outros parâmetros destacam-se a especificação do diretório para armazenamento de chaves autorizadas, encontrada sob o parâmetro *AuthorizedKeysFile*, a possibilidade de especificar uma ferramentas de gerenciamento de chaves por meio do parâmetro *AuthorizedKeysCommand* utilizando por exemplo *Lightweight Directory Access Protocol (LDAP)* (INFRASTRUCTURE..., 2020). Nota-se que a ferramenta especificada deve ter permissão de *root*, não escrita por grupo ou outros, isto é valor octal `700` e deve ser especificada por um *pathname* absoluto, de acordo com o manual do comando⁴² (SERMERSHEIM, 2020). Existe ainda um *script* que implementa um comando associado á esse arquivo de configuração, o `sshd_config` lendo os dados de configuração a partir desse diretório, também é possível editar diretamente o arquivo por `vim /etc/ssh/sshd_config`. Nota-se que linhas iniciadas por '#' ou linhas em branco são interpretadas como comentário (YLONEN, 2020b).

Estão disponível todos os protocolos *RFC* utilizados pelo *OpenSSH* no próprio site⁴³ bem como o índice contendo todos manuais para os comandos são disponibilizados pela própria *OpenBSD*⁴⁴.

2.2.6 Multi servidor *OpenSSH*

O arquivo `sshd_config` oferece a possibilidade de execução de mais de um servidor *daemon* *Openssh* na mesma máquina, prática constantemente explorada por empresas e

⁴²Disponível em <https://man.openbsd.org/sshd_config>

⁴³Disponível em <<https://www.openssh.com/specs.html>>

⁴⁴Disponível em <<https://www.openssh.com/manual.html>>

universidades (INFRASTRUCTURE..., 2020). Para, configurar *OpenSSH* como multi servidor são necessárias algumas etapas. Inicialmente, é copiado o arquivo de configuração do *daemon* por `cp /etc/ssh/sshd_config /etc/ssh/sshd2_config` de modo a ser utilizado pelo segundo *daemon*, edita-se esse arquivo de configuração copiado de modo a designar uma porta de aplicação diferente para essa instancia, para tal `sudo vim /etc/ssh/sshd2_config` ou `sudo sshd2_config` alterando o parâmetro *Port keyword* para um número alto desejado, por exemplo, 2222 (importante verificar que a porta atribuída não esteja sendo usada por uma outra aplicação) (MULTIPLE..., 2020).

Com o arquivo de configuração criado e devidamente alterado para essa nova aplicação é necessário tratar o arquivo de serviço especificado, ou *daemon* pelo *SSHD*⁴⁵, isto é, o arquivo responsável pela inicialização do *daemon*, toma-se a mesma abordagem copiando o arquivo e alterando os padrões de interesse `cp /usr/lib/systemd/system/sshd.service /etc/systemd/system/sshd2.service` (MULTIPLE..., 2020), os parâmetros de interesse do arquivo são mostrados abaixo na Figura 8.

Figura 8 – Especificações do *sshd_config*

```
1 user@localhost:~$ systemctl cat sshd.service
2 # /lib/systemd/system/ssh.service
3 [Unit]
4 Description=OpenBSD Secure Shell server
5 After=network.target auditd.service
6 ConditionPathExists=!/etc/ssh/sshd_not_to_be_run
7
8 [Service]
9 EnvironmentFile=-/etc/default/ssh
10 ExecStart=/usr/sbin/sshd -D $SSH_OPTS
11 ExecReload=/bin/kill -HUP $MAINPID
12 KillMode=process
13 Restart=on-failure
14 RestartPreventExitStatus=255
15 Type=notify
16
17 [Install]
18 WantedBy=multi-user.target
19 Alias=sshd.service
```

Fonte: Elaborado pelo autor

Inicialmente, altera-se a descrição do servidor de modo a distinguir da aplicação original, além disso, os parâmetros de inicialização e *alias* são alterados em *sshd2.service* a partir do padrão mostrado na Figura 8 para: `ExecStart=/usr/sbin/sshd -D -f /etc/ssh/sshd2_config $OPTIONS` e `Alias=sshd2.service` (MULTIPLE..., 2020). Desse modo, esse novo serviço será referenciado como *sshd2* pela determinação do *alias* e será associado o arquivo de configuração criado ao segundo *daemon* em sua inicialização, logo é possível customizar e alterar as determinações do segundo servidor independentemente do primeiro por meio de `sudo vim /etc/ssh/sshd2_config` (INFRASTRUCTURE..., 2020).

⁴⁵<<http://man.openbsd.org/sshd.8>>

É importante ainda, habilitar a porta *TCP* de comunicação atribuída nas configurações do segundo *daemon* `sudo sshd2_config` por meio de uma exceção no *firewall*, o que pode ser feito com `sudo iptables -A INPUT -p tcp --dport 2222 -j ACCEPT` (DEAL, 2004). Além disso, habilita-se e inicia-se o novo serviço depois das configurações com `systemctl enable ssh2.service ; systemctl start ssh2.service`, o carácter ";" foi utilizado de modo a inserir dois comandos em uma mesma linha no terminal, nota-se que devido à característica do comando *alias* utilizado na última linha do arquivo de configuração mostrado na Figura 8 é possível atribuir o novo *daemon* aos comandos que recebem serviços como parâmetro (NEGUS, 2020). Isto é, quando o cliente de *login* remoto *SSH* está autenticando um servidor o serviço será atribuído ao *host* o nome explicitado no valor *HostkeyAlias* ou então é atribuída a estruturação canônica (INFRASTRUCTURE..., 2020). Dessa mesma forma, como feito para outra aplicação é habilita inicialização do novo *daemon* automaticamente com o boot do sistema operacional por meio de `sudo systemctl enable sshd2` (YLONEN, 2020b). É interessante também criar um *symbolic link* com o binário *SSH*, cuja localização se dá pelo *pathname* `/usr/sbin/sshd` por meio de `sudo ls -s /usr/sbin/sshd /usr/sbin/sshd2` (MULTIPLE..., 2020). Dessa forma, uma vez que o primeiro servidor for atualizado o segundo servidor também será.

2.2.7 SSH Tunnel

Uma porta de rede é uma estrutura lógica responsável por identificar e especificar um processo ou serviço de rede, sendo a origem ou o destino do processo de endereçamento (KUROSE JAMES F.; ROSS, 2017). Nota-se que portas com número menor do que 1024 são atribuídas a aplicações gerais e bem conhecidas, de forma que são utilizados números elevado em novas implementações de modo a evitar conflito, além disso podem ser filtradas utilizando o *firewall* (DEAL, 2004).

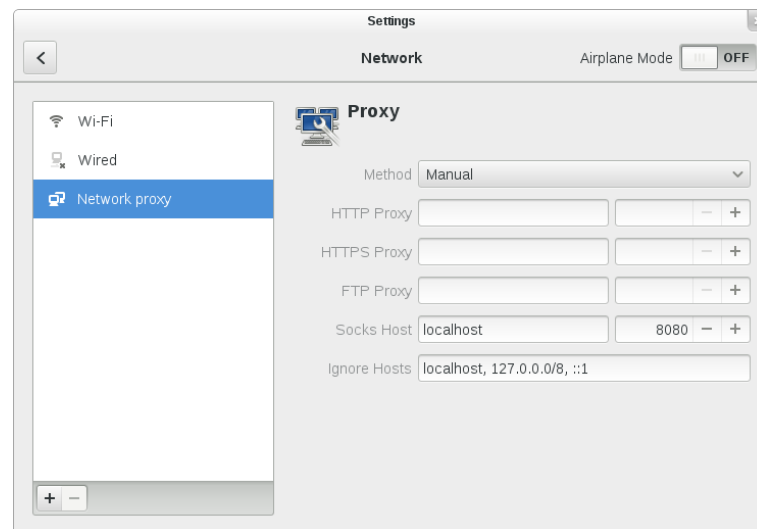
O tunelamento *TCP/IP*, ou encaminhamento de porta *SSH* é um mecanismo usado para direcionamento de tráfego de portas de aplicação da máquina do cliente para a máquina do servidor (KUROSE JAMES F.; ROSS, 2017). Sendo utilizada para mascaramento de *IP* e principalmente como *proxy* para acessar uma rede interna de um servidor por meio de um cliente em outra localidade geográfica (INFRASTRUCTURE..., 2020), de modo que um *proxy* atua como um nó intermediário da rede entre o computador do cliente e a *internet* e dessa forma o tráfego é visto como gerado pelo servidor *proxy* (WHITE, 2018). Além disso, se o *firewall* está configurado de modo a permitir tráfego por meio da porta padrão 22 mas bloqueia acesso a outras portas a comunicação entre dois usuários ainda é possível se redirecionar a comunicação sob uma conexão *SSH* estabilizada.

Dessa forma, utiliza-se no projeto o encaminhamento de porta dinâmico por meio de uma conexão *SSH* de modo a acessar os computadores da rede interna do sistema de *cluster* (CHAPPLE, 2020). Por meio do *Socket Secure (SOCKS5)*, um protocolo de rede

que facilita a comunicação com servidores atrás de *firewalls*, por rotear o tráfego de rede por um *proxy* (KUROSE JAMES F.; ROSS, 2017). Dessa forma, é possível se conectar diretamente à rede local do laboratório LPS por qualquer distribuição linux em qualquer dispositivo com acesso ao *SSH*.

Um servidor *proxy SOCK* cria um protocolo de conexão *TCP* para outro servidor atrás de uma *firewall* do lado do cliente, então efetuam-se troca de pacotes de rede entre cliente e servidor (MACH, 2020). Nota-se que o *SOCK* é protocolo de camada 5 no modelo *OSI*, de modo que não pode ser usado no tunelamento de protocolos operando abaixo da quinta camada (KUROSE JAMES F.; ROSS, 2017). Logo, tornando a conexão imune a ataques de terceiros se utilizando de ferramentas conhecidas como *Address Resolution Protocol (ARP)* e *Network Mapper (NMAP)* (NMAP..., 2020). Além disso, *SOCK* está entre *Secure Sockets Layer (SSL)* e *TCP/UDP* permitindo requisições do tipo *Hypertext Transfer Protocol (HTTP)*, *Hypertext Transfer Protocol Secure (HTTPS)*, *Post Office Protocol version 3 (POP3)*, *Simple Mail Transfer Protocol (SMTP)* e *FTP*, de modo que é possível utilizar *e-mail*, *web browsing*, *peer-to-peer sharing* sem a necessidade de uma *Open Source Connection Protocol (OpenVPN)* (PATIL, 2020).

A sintaxe utilizada `ssh -D 8080 -C -N usuario@IPSERVER` é empregada no cliente *SSH* utilizando *linux*, explicitando-se como criar um servidor *proxy* por *socket* a ser executado na maquina do cliente e então autentica-lo ao nó principal do *cluster* por meio do *gateway* responsável por rotear o tráfego internamente para o servidor local do sistema de cluster LPS (PATIL, 2020). Especifica-se o encaminhamento de porta dinamicamente por meio do parâmetro `-D`", alocando-se um *socket* para checar conexões em uma porta especifica `"p=8080"`. Dessa forma, é necessário configurar *browser* no cliente de modo a redirecionar o tráfego a partir da rede local no servidor *SSH*, no entanto, esses parâmetros são atribuídos globalmente nas próprias configurações de rede do *linux* em conexão por *proxy* manual, selecionando opção por *socket* com *IP* 127.0.0.1 (*IP* de *lookback*) e porta de aplicação 8080 como mostrado na Figura 9, de modo que o cliente *SSH* de e o servidor *proxy* virtual estão sendo executados na mesma máquina. Dessa forma, com endereço de endereço *IP* de *loopback*, com conexão estabelecida para a porta `"p"` o tráfego é encaminhado pelo canal *SSH* e é estabelecido um servidor *tunnel* acessando a rede local do servidor *SSH* e por conseguinte toda a *localhost* do *cluster* (PATIL, 2020).

Figura 9 – Especificações do *Proxy* no Sistema Operacional

Fonte: Elaborado pelo autor

Nota-se que p normalmente é atribuído em conexões de *proxy* e tunelamento a porta 8080, observa-se que valor escolhido apenas por questões derivativas de especificidade de porta de aplicação *web* cujo número é 80 (KUROSE JAMES F.; ROSS, 2017). No entanto, qualquer porta pode ser especificada para atuar como *tunnel*, recomenda-se apenas que seja um número maior do que 1024 de modo a evitar eventuais conflitos com outras aplicações (PATIL, 2020).

2.2.8 Pacotes e dependências

Diferentes distribuições de sistema operacional podem ser estruturadas sob a arquitetura de *kernel* com diferente sintaxe para gerenciamento de pacotes (NEGUS, 2020). Uma vez que servidores ou computadores em *linux* podem se utilizar de diferentes métodos na atribuição de *software* no desenvolvimento do *kernel*, afetando a execução de um determinado programa globalmente (SHOTTS, 2019). Dessa forma, de modo a resolver tal interoperabilidade pacotes normalmente incluem dependência para sua execução sendo necessário atribuições de gerenciamento de pacotes internamente no sistema operacional (MACH, 2020).

Sob essa perspectiva, o *Package Management System (PMS)* é implementado no sistema de modo a controlar instalação de software e aplicações bem como as bibliotecas necessárias para sua execução (MACH, 2020). O gerenciamento de pacotes se dá por meio de repositórios de *softwares* sendo essa a forma mais eficiente de se instalar e manter atualizados *softwares* em um sistema (BLUM, 2020). Ora, como pacotes de *softwares* são armazenados em um servidor, referido como repositório, são acessados por uma conexão com a internet previamente estabelecida no sistema *linux* (Wang et al., 2009). É comum

um software ou utilitário apresentar dependências sendo necessário instalar outros pacotes para a execução adequada (SHOTTS, 2019)

Os pacotes dos sistemas baseados em *Debian* como *Ubuntu*, *linux Mint*, *ElementaryOs* usam na estruturação básica do utilitário *PMS* o comando *Debian Package (DPKG)*, responsável por gerenciar pacotes *.deb*. Enquanto os pacotes dos sistemas baseados em *Red Hat* como *REHL*, *Fedora CentOS* e *Scientific Linux* usam na estruturação básica do utilitário *PMS* o *Red Hat Package Manager (RPM)* (SHOTTS, 2019). Ora, alguns comandos são responsáveis pelo gerenciamento dos pacotes, isto é, por listar, instalar e remover software (MACH, 2020). É possível obter os principais pacotes das principais distribuições na iniciativa *PKGS*⁴⁶ (BOTH, 2020).

O *PMS* padrão a ser utilizado é o *RPM* de acordo com *Linux Standard Base (LSB)*, projeto aderido por varias distribuições de *linux* sob uma organização estrutural com objetivo de padronizar os *softwares* bem como estruturas do sistema, atualmente se encontra-se na especificação *LSB5*⁴⁷ (LINUX..., 2020). Inclusive, existe um comando associado que ao ser digitado no terminal exibe informações sobre distribuição específica instalada, com sintaxe `lsb\release` de forma que é exibido o número da versão, nome do código da versão e do distribuidor (LSB, 2020).

Para distribuições *RHEL* e seus derivados existem ainda requirições para instalação do *software netdata* que não são necessárias em outras distribuições, entre elas a instalação do *Extra Packages for Enterprise Linux (EPEL)* (LEARN..., 2020). Trata-se de um pacotes de código aberto selecionados a partir da distribuição *Fedora*. Devido á tomadas de diretrizes distintas *RHEL* e *fedora* não compartilham os mesmos conteúdos uma vez que uma distribuição atua em âmbito corporativista e outra é uma distribuição de código livre. Dessa forma, a própria distribuição *RHEL* quanto distribuições baseadas, excetuando-se o *fedora*, carregam essa limitação de dependência, é o caso por exemplo de *CentOS*, *scientific linux* entre outros (EPL, 2020). Desse modo, além dos pacotes de manipulação da ferramenta básica de gerenciamento de pacotes em que a arquitetura do sistema operacional é baseada, são necessários alguns outros pacotes para implementação do *software netdata* nessas distribuições (NETDATA..., 2020c).

EPEL é um *Special Interest Groups (SIG)*, ou grupo de interesse do próprio projeto *fedora*, responsável por fornecer e integrar diferentes ferramentas a partir do *fedora* para os outros sistemas operacionais baseado em *RHEL*, apesar da próprio *RH* não ter participação ou fornecer suporte ou correção de erros (EPL, 2020). Entre o conteúdo do pacote são disponibilizados ferramentas de redes, ferramentas relacionadas ao sistema por meio de *sysadm*, monitoramento e programação. O método de instalação para as distribuições do ecossistema suportado bem como o repositório com a última versão podem

⁴⁶Disponível em: <<https://pkgs.org>>

⁴⁷Disponível em: <<https://www.linuxfoundation.org>>

ser encontradas no próprio site do projeto fedora ([PROJECT, 2020](#)). Além disso, existe outro pacote cuja dependência é o próprio *EPL* e é necessário nesse contexto, trata-se do *Okay* ([NETDATA..., 2020c](#)). Atua de forma similar completando algumas dependências necessárias para executar o *software netdata* nesses sistemas operacionais, principalmente com relação ao *package libuv version 1*⁴⁸ e a partir do próprio repositório *Okay*⁴⁹ é instalado com `yum install epel-release`.

2.3 Cluster

O *cluster* Beowulf pertence a uma classe de sistema paralelo ou distribuídos sendo uma implementação de otimização por meio de computação paralela. Ora, muitas tarefas computacionalmente complexas podem ser executadas mais rapidamente com a tarefa distribuída por diferentes processadores, dessa forma e essa topologia consiste em um agrupamento de computadores pessoais trabalho juntos como um único computador de recurso integrado. Dessa forma, o Beowulf pertence a uma classe de sistema paralelo ou distribuídos, consistindo de um agrupamento de computadores pessoais trabalho junto como um único computador de recurso integrado ([DAHILI, 2001](#)). Em 1994, *Thomas Sterling* e *Don Becker* do Centro de Voo Espacial *Goddard* da *NASA* encontraram uma solução para os problemas envolvendo os sistemas *High-performance computing (HPC)* comerciais, principalmente com relação ao elevado custo de desenvolvimento de hardware e software para época ([ADAMS; VOS, 2002b](#)). O problema foi resolvido por meio da conexão de 16 computadores 486-DX4 por meio de *Ethernet* e se utilizando da instalação de software livre (*linux, MPI, PVM*) para obter o comportamento de um multiprocessador, obteve-se assim uma velocidade significativa a uma pequena fração do preço de um supercomputador equivalente, nomeando-o como *Beowulf*, como citado no artigo original ([STERLING et al., 1995](#)).

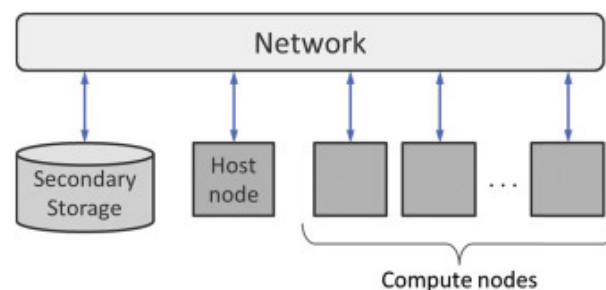
O processo de montagem de um cluster envolve a criação e configuração de uma rede *LAN*, normalmente baseada em *FastEthernet* ([THEPUATRAKUL, 2010](#)). De modo que cada computador da rede é chamada de um nó do cluster *cluster node*, além disso cada nó tem *linux* instalado e operando. Alguns compiladores devem ser instalados para o seu funcionamento, como *MPICH*, uma implementação de alto desempenho e amplamente portátil da Interface de Passagem de Mensagens *MPI* para processamento distribuído, além do *SSH* ou *Secure-Shell*, um protocolo que se utiliza de encriptação para estabelecer e assegurar a ligação entre um cliente e um servidor, de modo que é utilizado um conjunto de utilitários de rede (*OpenSSH*) baseado nesse (*SSH*), capaz de fornecer um canal de comunicação através de uma rede para execução remota. Além disso, o *NFS* ou é essencial para o compartilhamento do sistema de arquivos ([DATTI; UMAR; GALADANCI, 2015](#)).

⁴⁸Disponível em: <<https://github.com/libuv/libuv>>

⁴⁹Disponível em: <<http://repo.okay.com.mx>>

O principal componente de um sistema é referido como o "nó" e inclui a maior parte dos componentes ativos e o trabalho computacional que compõem o sistema agregado chamado *cluster*, o nó por si só é um computador funcional e completo (STERLING; ANDERSON; BRODOWICZ, 2018). Dessa forma, a capacidade de desempenho de um cluster é essencialmente a capacidade agregada de todos os nós de computação combinados. Os nós em combinação com a rede e o armazenamento secundário compreendem a estruturação de um *cluster* completo como mostrado na Figura 10. Os nós em uma topologia original necessitam apenas da capacidade de armazenamento necessária para a execução do código (COOK, 2013). No entanto, um armazenamento secundário é associado a uma estruturação de *cluster* de forma a fornecer estabilidade no armazenado de arquivos e diretórios, códigos e programas, bem como os dados associados á aplicações e programas que são executados no *cluster* (STERLING DANIEL F. SAVARESE, 1999).

Figura 10 – Diagrama de blocos de uma estrutura canônica de *cluster*



Fonte: (STERLING; ANDERSON; BRODOWICZ, 2018)

2.4 Software Netdata

Com o aumento de recursos computacionais aumentaram-se o número de variáveis associadas ao desempenho do sistema, de forma que para o gerenciamento de um sistema como o de *clustering* são necessárias mais ferramentas visuais otimizadas que potenciem a capacidade da visão humana de analisar grandes quantidades de informação simultaneamente (Tschinkel et al., 2015).

A maior parte das soluções de monitorização existentes explicitam apenas localmente quando ocorre um problema, sendo necessário a utilização de diferentes ferramentas e manipulação no terminal do sistema operacional para encontrar a causa e só então tentar implementar uma solução. Opta-se pela implementação de uma ferramenta que gere informações a todo momento em tempo real de que modo a manter uma base de dados para compreender a razão pela qual uma anomalia aconteceu e resolve-la (YIGITBASIOGLU; VELCU, 2012).

Dessa forma, para o gerenciamento do sistema opta-se pela implementação de uma ferramenta que promove integração entre as tecnologias monitoradas de modo a distinguir

como as métricas se correlacionam e suas funcionalidades no sistema (LEARN... , 2020). Logo, a ferramenta deve implementar diretivas para auxiliar o usuário a compreender globalmente o que está sendo monitorado, fornecendo informação e descrição das métricas de forma gráfica e interativa (YIGITBASIOGLU; VELCU, 2012).

Por ser uma ferramenta grátis e de código aberto, optou-se pela implementação do *software netdata*, cujos códigos, atualizações e implementações estão disponíveis no *git* do próprio desenvolvedor ⁵⁰.

O *netdata* atua em tempo real, isto é, recolhem-se todas as métricas referente ao sistema a cada segundo e com latência menor do que um milissegundo na disponibilização das alterações para visualização (STREAMING... , 2020a). Ora, a visibilidade praticamente imediata atribuí maior granularidade no gerenciamento de dados com a detecção de picos (*spikes*) em determinados parâmetros que influenciam diretamente na infra-estrutura e funcionalidade do sistema de *clustering*, avalia-se constantemente desde do uso de *CPU* e memória até a perda de pacotes em cada nó da rede (LEARN... , 2020). Dessa forma, o *software* por meio de alarmes e avisos permite tomada de decisões mais rápidas, direcionadas e tecnicamente embasadas sobre o estado e o desempenho do sistema, possibilitando verificar imediatamente alterações no sistema pelo gerenciamento das aplicações sendo executadas. Assim, é especialmente útil em implementações de ambientes de nuvens contemporâneos, onde o desempenho da infra-estrutura não é linear nem previsível, detectando rapidamente alguma anormalidade (FERREIRA et al., 2013). Além disso, apresenta aplicações também em sistemas de computação paralela como *clustering* onde são processados muitos dados de forma paralela e simultânea como descrito na Sessão 2.3.

O conceito de painel de controle melhora a tomada de decisões através da visualização dos processos e ajuda a determinar onde os processos desempenham funcionalidades de acordo como esperado e onde podem ocorrer problemas potenciais (SEDRAKYAN; MANNENS; VERBERT, 2019). No entanto, o conceito por trás da ferramenta *netdata* não está exclusivamente associado ao conceito de uma interface estática, pré configurada e finita. Mais do que um painel de controle o *software* é uma ferramenta de solução de problemas. Uma vez que a ferramenta *netdata* é muito mais rápida, flexível e dinâmica, do que um painel de controle, implementando meios para explorar, gerenciar e trabalhar com todas as métricas de uma forma significativa⁵¹ (LEARN... , 2020). Dessa forma, quando no projeto o termo *dashboard* for referenciado este é utilizado no contexto de uma implementação *netdata*, isto é, refere-se ao painel interativo em tempo real, altamente granular e com gráficos personalizados próprios da ferramenta.

Os recursos para operação do *software* não são elevados, sendo inclusive utilizados em dispositivos com *hardware* limitado para aplicabilidade em internet das coisas (IOT... ,

⁵⁰Disponível em: <<https://github.com/netdata/netdata>>

⁵¹Disponível em: <<https://learn.netdata>>

2020). Ora, desde das diretivas de desenvolvimento inicial da ferramenta é priorizado o baixo uso de memória *RAM* de modo a interferir o mínimo possível no sistema (DBENGINE, 2020). Dessa forma, não existe a necessidade de centralizar as métricas em um único sistema com maior capacidade computacional e assim cada nó do sistema de *clustering* é responsável por recolher métricas, manter sua própria base de dados, acionar alarmes, e construir *dashboard* localmente, e então só posteriormente enviar dados pela rede para o nó principal e compôr a construção da *dashboard* em um único *display* (LEARN..., 2020).

Dessa forma, por meio de agentes que monitoram a saúde do sistema e suas aplicações, a instalação automática do *netdata* já configura previamente dezenas de alarmes e alertas para o quando o sistema começa a agir de forma não usual. Os alarmes podem ser configurados de acordos com especificidades de aplicação por meio da sessão *health* no arquivo de configuração principal da aplicação (*netdata.conf*) (HEALTH..., 2020).

O site oficial da aplicação dispõe de diferentes sessões informativas sobre a ferramenta ⁵². Além disso, são encontradas diretrizes de configuração submetidas pelos próprios desenvolvedores e colaboradores (OPEN..., 2020b). De forma que todas documentações referentes á ferramenta estão disponíveis e organizadas em sessões quanto á aplicabilidade ⁵³. Ainda são disponibilizados guias didáticos com diferentes níveis de complexidade sobre o uso da ferramenta na sessão *guides* sob a iniciativa *learn netdata* ⁵⁴. Diante de comportamentos não esperados como *bugs* e demais dificuldades técnicas o espaço denominado *issues* no próprio *GitHub* da ferramenta é disponibilizado para solução de problemas, contando com auxilio tanto de desenvolvedores e colaboradores quanto da própria comunidade ⁵⁵. Ora, a política de código aberto infere que cada problema e dificuldade abordada é uma potencial valiosa contribuição á ferramenta (OPEN..., 2020a).

2.4.0.1 Database

Com o projeto inicial armazenando a base de dados de métrica na memória *Random Access Memory (RAM)*, foi desprendido um grande esforço da parte dos desenvolvedores de modo a manter processo extremamente otimizado viabilizando implementações tanto em aplicabilidades com limitação de *hardware RAM* quanto aplicações mais robustas. Por exemplo, para um taxa de coleta de métricas de 1000 métricas/segundo e armazenamento histórico de métrica configurado para uma hora a ferramenta utiliza 14.4MB de *RAM*, dessa forma para um dia de métricas armazenadas nessa taxa seria alocado 345MB de *RAM*. Para sistemas mais criticos cujas implementações necessitam de maior granularidade e constante monitoramento de estabilidade aumentaria-se a taxa de coleta, de modo que

⁵²Disponível em: <<https://www.netdata.cloud/>>

⁵³Disponível em: <<https://learn.netdata.cloud/docs/agent>>

⁵⁴Disponível em: <<https://learn.netdata.cloud/guides>>

⁵⁵Disponível em: <<https://github.com/netdata/netdata/issues>>

de acordo com o desenvolvedor para uma taxa de 100000 métricas por segundo seria utilizado 1.7GB de *RAM* para armazenar uma hora de métrica (DBENGINE, 2020).

Contemporaneamente armazenar dados exclusivamente na memória *RAM* apresenta uma limitação para certas aplicabilidades, uma vez que devido às suas características e volatilidade a *RAM* pode ser empregada de forma mais eficiente do que exclusivamente para armazenamento (M., 2005). Devido á crescente demanda por armazenar maiores quantidades de métricas por maior período de tempo (dias, semanas ou meses) (AKHILA; GANESH; SUNITHA, 2016), em um contexto onde a quantidade de dados disponíveis é tida como fator determinante para tomada de decisões em diferentes áreas como medicina, mercado financeiro e meio corporativo, o aumento do espaço amostral agrega assim capacidade preditiva na diretriz de desenvolvimento em diferentes âmbitos (SPIVAK et al., 2018). Como por exemplo, técnicas distribuídas de armazenamento de maiores quantidades de dados podem ser utilizadas em implementações mais eficientes para outras técnicas de processamento de dados (J. et al., 2020). Dessa forma, com maior capacidade de armazenamento disponível é definida a eficiência de armazenamento como a capacidade de armazenar e gerenciar dados alocando menos recursos sem impactar na performance, resultando diretamente em um menor custo operacional (SNIA. . . , 2020). Como definido pela *Storage Networking Industry Association (SNIA)*⁵⁶.

O modo de operação de memória pode ser selecionado editando o arquivo de configuração principal do software *netdata.conf* em */etc/netdata* (DBENGINE, 2020). Entre as opções são possíveis:

- *Dbengine*
- *RAM*
- *SAVE*
- *MAP*
- *NONE*
- *ALLOC*

O modo de memória *dbengine* foi implementado a partir da versão v1.15.0, de modo a fornecer na sua estruturação de base de dados a opção de armazenamento de métricas comprimidas, principalmente por um período de tempo maior do que seria possível utilizando exclusivamente a memória *RAM* disponível no sistema, atribui-se a esse modo a nomenclatura *dbengine*, referente a *data base engine* (RELEASE. . . , 2020). A implementação funciona de modo análoga á uma base de dados tradicional, armazenando-se o histórico

⁵⁶Disponível em: <<https://www.snia.org>>

de métricas no disco principal *Hard Drive (HD)* em um formato comprimido, difere-se por se utilizar das características associadas à memória tipo *RAM*, como volatilidade e acesso rápido (AGENT..., 2020). Ora, a técnica de indexação de memória referente ao endereçamento deve ser feita de forma rápida e otimizada, desse modo é alocado previamente uma quantidade de *RAM* tanto para indexação quanto para a técnica de *caching*, isto é, reserva-se temporariamente na *RAM* espaço para informações e dados que são constantemente acessados pelo programa, de modo que quando a informação é requisitada em um outro momento não é necessário gerá-la novamente (M., 2005).

De acordo com a desenvolvedor o uso da *RAM* nesse modo equivale a 3% do espaço requerido pelos arquivos no disco (DBENGINE, 2020). O valor está intrinsecamente relacionado à quantidade explicitada nos arquivos de configuração do parâmetro *page cache size* referente ao tamanho máximo reservado para o *cache* em MiB, por padrão é alocado 32 MiB (*Mebibyte*) (AGENT..., 2020). Nota-se que o prefixo "*Mebi*" está associado ao multiplicador derivado do sistema internacional de unidades (*mega*) enquanto o sufixo da própria palavra "*Mebi*" remete ao binário, em base 2. Assim, refere-se $1 \text{ MiB} = 2^{20} \text{ bytes}$. Essa conotação utilizado em informação digital está associada de forma análoga à contra parte decimal, isto é, enquanto $1000B$ é $1KB$ no SI, e $(1000)^2B$ é $1MB$. Utiliza-se $1024B$ como $1KiB$ e $(1024)^2B$ como 1 MiB . Ora, como $2^{(10)}B = 1024B$ definiu-se o sufixo *iB* de modo a trabalhar na base binária. Logo, $1 \text{ MiB} = 2^{20} \text{ bytes} = 1048576 \text{ bytes} = 1024^2 \text{ bytes}$, mantendo-se a proporcionalidade entre as grandezas inerente ao SI, isto é, com $1 \text{ MiB} = 1024 \text{ kibibytes}$ (M., 2005).

Além disso, a implementação em *dbengine* permite alterar a frequência de atualização utilizada na coleta de dados por meio do parâmetro *update_every* sem acometer dados previamente coletados e armazenado (DBENGINE, 2020).

Para seleção em *RAM* os dados são armazenamento diretamente na memória *RAM* e nunca acessado por disco, o mapeamento de memória se dá diretamente por *nmap* (AGENT..., 2020). Ora, o *nmap* é um conjunto de protocolos agregados sobre *Portable Operating System Interface (POSIX)* (NMAP..., 2020), pertencente à uma classe de padronizações especificados pela própria *IEEE*⁵⁷ de modo a manter a compatibilidade entre variantes de *UNIX* e outros sistemas operacionais (GIFT, 2008). Por exemplo, pela *POSIX* é definida a aplicação da interface de programação (*API*) (GAO et al., 2021).

Dessa forma, o *nmap* mapeia dispositivos e arquivos na memória por meio de requisições programática de serviços executados no *kernel* do sistema operacional atendendo às especificações da padronização *POSIX* (GIFT, 2008). Isto é, a requisição *mmap system call* permite a leitura e escrita em um arquivo como se estivesse sendo acessado um vetor ou *array* diretamente na memória (DBENGINE, 2020). Ora, dados são essencialmente vetores de *bytes* armazenados nos arquivos de sistema, dessa forma, a chamada da requisição *nmap*

⁵⁷Disponível em: <<https://www.ieee.org/standards/index.html>>

está diretamente relacionada á forma como o sistema operacional *linux* gerencia arquivos bem como se dá o próprio endereçamento de memória (M., 2005).

Além de utilizar o *nmap* essa implementação de base de dados suporta *Kernel Samepage Merging (KSM)*, isto é, um aspecto da estruturação do *kernel* que permite ao *Kernel-based Virtual Machine (KVM)* o compartilhamento de blocos de memórias idênticas entre diferentes processos ou máquinas virtuais no mesmo servidor (AGENT..., 2020). Ora, um *host KVM*, referente á *Kernel-based Virtual Machine*, é uma implementação de virtualização em código aberto (KVM, 2020). Dessa forma, por ser a cerne da virtualização em *linux* o *KVM* é composto normalmente por diferentes instâncias de máquinas virtuais, com diferentes interfaces de *hardware* virtualizadas (HYPERVISOR, 2020). Logo, blocos de memória podem apresentar redundância contendo o mesmo conteúdo, como por exemplo diretrizes de operação do próprio sistema operacional que são executadas para cada virtualização. Com a implementação do *KSM* esses blocos de memória passam a ser identificados de forma unívoca em uma localização, otimizando o gerenciamento de memória por meio de um maior esforço de recursos *CPU* para identificação e manutenção da coesão desses blocos (VIRTUALIZATION, 2020). A implementação *KSM* assim mantém semelhança junto ao processo de *Data deduplication* (AKHILA; GANESH; SUNITHA, 2016).

No modo *Save* enquanto a aplicação está sendo executada as métricas também são armazenadas somente na *RAM* do sistema, no entanto, difere-se quanto á capacidade de salvar no disco métricas armazenadas na *RAM* em uma eventual reinicialização, ou então carregar métricas a partir do disco quando o sistema for eventualmente iniciado. Utiliza-se também de *nmap* e *KSM* (DBENGINE, 2020).

No modo *Map* as métricas estão em um arquivo de memória mapeado, isto é, um segmento de virtualização de memória que está alocado á uma correlação direta do tipo byte por byte (RESOURCER..., 2020). Esse modo atua de forma semelhante á técnica de *memory swap* em *linux*. Ora, existe uma limitação física na disponibilidade de *RAM* no sistema, de modo que quando é atingido o limite do processamento o sistema responde por meio do conceito de memória virtual. No entanto, em *linux* quando o limite é atingido o sistema operacional aloca memória de uma unidade de armazenamento secundária para armazenar o conteúdo inativo que está ocupando a *RAM* do sistema, de modo a liberar espaço e assim executar o processo atual. Dessa forma, o espaço alocado pelo HD é chamado de memória *Swap* (SWAP..., 2020).

Dessa forma, o *kernel* do *linux* interpreta os dados escritos nessa virtualização de memória como blocos passíveis de alocação por *swap* e automaticamente os direciona ao disco rígido. A frequência de atualização é exatamente a mesma de *swaps* característico gerenciados pelo sistema operacional, uma vez que o *kernel* do *linux* se utiliza do mesmo algoritmo de *swaps*. Esse modo utiliza-se de *nmap* no entanto não permite implementação

em virtualização com *KSM* (DBENGINE, 2020).

O modo de memória *None* não possibilita a manutenção de uma base de dados próprias e modo *alloc* atua de forma semelhante ao modo *RAM*, no entanto todos os *bits* da memória alocada são zerados (CALLOC, 2020). Esse é o modo de operação atribuído quando a configuração de seleção selecionada não seja possível, excetuando-se a opção *None* (DBENGINE, 2020).

2.4.1 Streaming

A ferramenta de *streaming* presente no *software netdata* é uma implementação de comunicação pela rede *LAN* ou *WLAN* interna integrada pelo próprio *software* e permite a replicação das métricas gerados e seu encaminhamento, isto é, uma base de dados gerada em um nó é replicada e encaminhada para outra aplicação *netdata* por meio da rede interna, dessa forma a ferramenta de *streaming* distingue os nós de uma rede baseando-se nos conceitos de que os nós que enviam métricas são chamados de *child nodes*, enquanto os nós que recebem métricas são chamados de *parent nodes*. Além disso, aos nós também pode ser designadas a função intermediária de recolhimento de métricas de um outro *child node* e encaminhamento a posteriori para um dos *parent node*, essa última configuração recebe a nomenclatura de *proxie* (STREAMING..., 2020a).

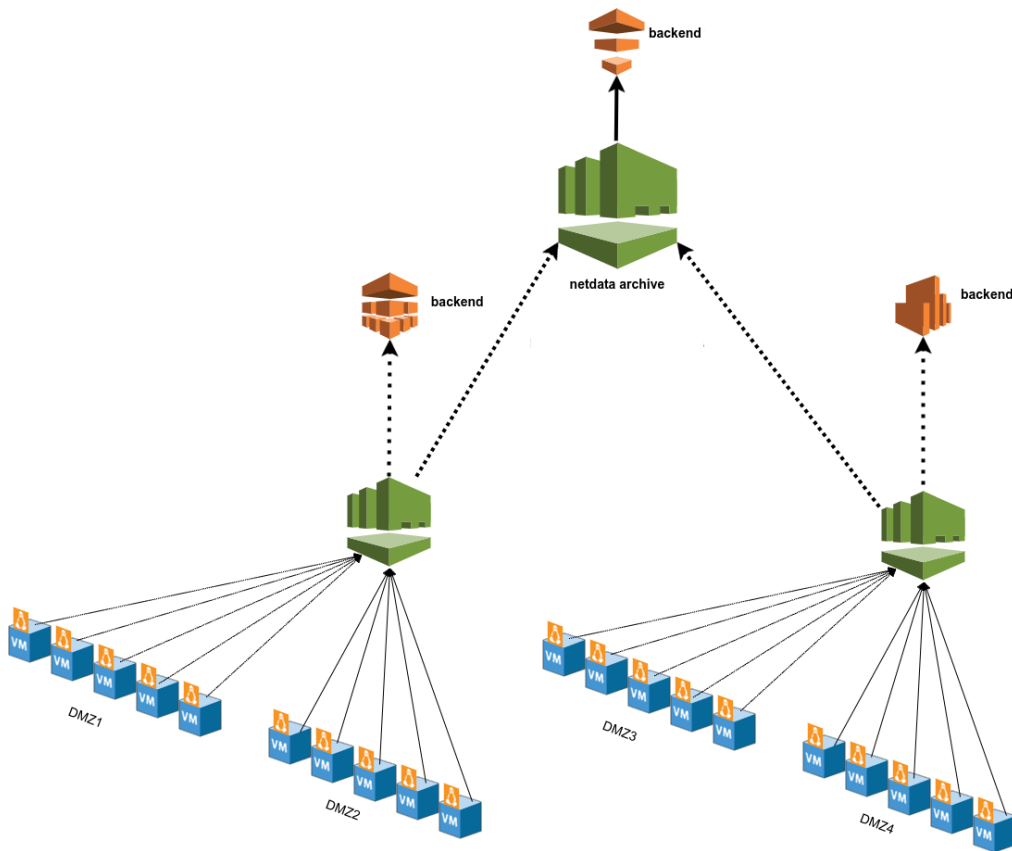
No entanto, a ferramenta precisa ser configurada para cada rede particular de modo a estabelecer a comunicação e tal configuração se baseia nos conceitos de que os nós que enviam métricas são chamados de *child nodes*, e os nós que recebem métricas são chamados de *parent nodes*, além disso aos nós também pode ser designadas a função intermediária de recolhimento de métricas de um outro *child node* e encaminhamento para um dos *parent node*, essa configuração recebe a nomenclatura de *proxie* (STREAMING..., 2020a).

Tabela 3 – Possíveis configurações dos nós no *netdata*.

Nó	Modo de memória	Modo <i>web</i>	<i>stream</i>	<i>dashboard</i>
Apenas coletor	nenhum	nenhum	ativado	impossível
Apenas <i>proxie</i>	nenhum	nenhum	ativado	impossível
<i>Proxie</i> com Db	selecionar	estático	ativado	possível
nó central	salvar	estático	desativado	possível

Fonte: Adaptado de (STREAMING..., 2020a)

De forma adicional, quando existe necessidade de implementações de topologias mais complexas o *netdata* permite maior seletibilidade na disposição dos nós e tratamento dos parâmetros, como por exemplo na topologia da Figura 11.

Figura 11 – Topologia *streaming netdata*

Fonte: Adaptado de (STREAMING..., 2020a)

Na Figura 11 existem 3 coletores de métricas, o coletor 1 um recebendo métricas da *Demilitarized Zone (DMZ)*, *DMZ1* e *DMZ2* enquanto o coletor 2 recebe métricas da *DMZ3* e *DMZ4*. Além disso, ambos coletores mantêm a base de dados referentes às métricas coletadas das respectivas *DMZ* inclusive exibindo-as em uma interface de um servidor referido como *backend* tal como *prometheus* ou *grafite*, uma lista de *backend* suportados pelo software *netdata* pode ser encontrado nas documentações do software⁵⁸. No entanto, esses coletores também funcionam como servidores *proxie*, isto é, encaminham essas métricas para um terceiro coletor, usualmente responsável por manter uma base de dados maior e por mais tempo, além de exibir as métricas de todos os nós associados à todas *DMZ* localmente ou por meio de um *backend* (STREAMING..., 2020a). Ora, em uma estruturação desse tipo apenas os parâmetros referentes ao modo de memória e *web* não são suficientes para descrever cada comunicação entre servidor, *host* e *proxy* de modo que foi implementado um parâmetro cuja manipulação se dá no mesmo arquivo de configuração de *streaming* e funcionalidade está associada à identificação da aplicação *netdata* geradora de métricas em cada nó, tal parâmetro é um *Globally Unique Identifier (GUID)* aleatório gerado quando a aplicação é instalada no nó, de modo que cada aplicação do *netdata* instalada tem um parâmetro diferente associado e localizado no diretório

⁵⁸Disponível em <<https://learn.netdata.cloud/docs/export/external-database>>

/var/lib/netdata/registry/netdata.unique.id permitindo a sua distinção na rede, define-se ainda que o identificador global único não é alterado quando um recurso ou aplicação é atualizado (C., 2015).

2.4.2 API

A interface de programação de aplicações *Application Programming Interface (API)*, essencialmente é um conjunto de rotinas e protocolos de comunicação, definindo interações entre *softwares*. É uma ferramenta popularmente utilizada para permitir a comunicação entre processos diferentes sendo executados em um nó, ou mesmo atuando em diferentes *host* (GAO et al., 2021).

Uma chave *API* é um identificador global único *GUID*, isto é, um código passado como parâmetro em diferentes aplicações que estão sendo executadas em um computador ou em uma rede, de modo a facilitar a comunicação por meio da identificação e autenticação (STEVENS BILL FENNER, 2003).

O sistema *linux* apresenta implementado um comando gerador universal de identificadores únicos *Universal Unique Identifier (UUIDs)*, com sintaxe `uuidgen`, de modo que é possível gerar uma chave de *API* aleatória e utiliza-la em uma aplicação (SHOTTS, 2019), um exemplo de chave *API* gerada dessa forma é: 34319e67-1852-46ba-bd8c-6cf6ce6f2c0e. Salienta-se que essa é uma *API* gerada aleatoriamente e será utilizada exemplificar a implementação do projeto nos arquivos de configuração, não sendo a chave de configuração da ferramenta *netdata* no laboratório LPS.

Na configuração do *netdata* é necessário atribuir uma chave *API* dependendo da classificação do nó, de acordo com a Sessão 2.4.1, sobre *streaming* (STREAMING..., 2020a). Dessa forma, atribui-se uma chave *API* ao nó principal, no caso LPS00, por meio do comando `uuidgen` e uma outra chave *API* aos nós secundários. O *netdata* permite atribuição distintas para cada *child node*, de modo que comuniquem-se separadamente com o nó principal, no entanto para a aplicação da análise de métricas em um *cluster* uma chave *API* para o nó principal *parent node* e a mesma *API* para todos *child node* (STREAMING..., 2020a).

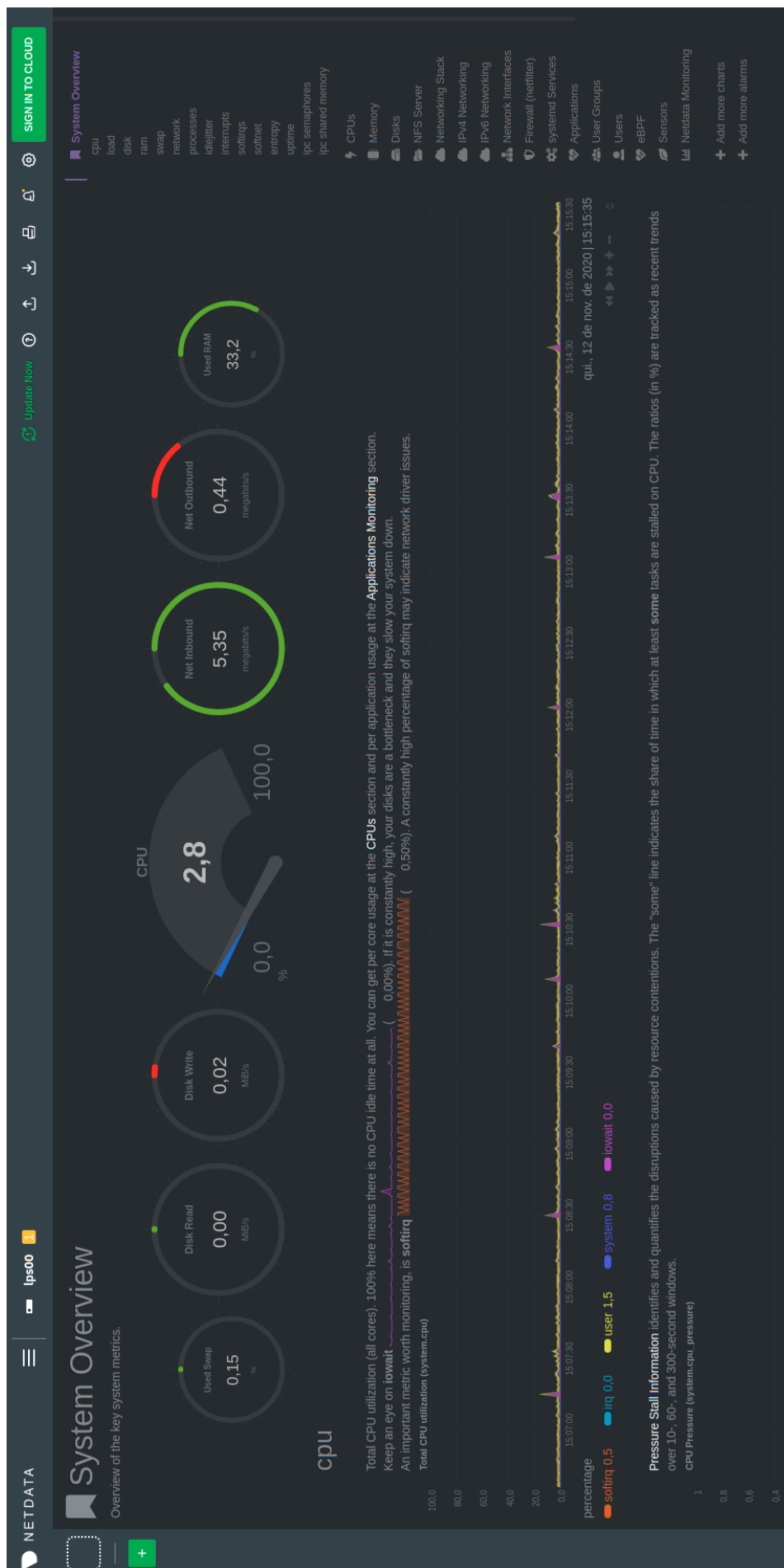
2.4.3 Dashboard

Como descrito na introdução á ferramenta *netdata* o conceito de análise de métrica está intrinsecamente relacionado ao conceito de *dashboard*. Descreve-se assim nessa sessão suas características e funcionamento.

A *dashboard* padrão do *software netdata* é mostrada na Figura 12 e é onde se dá a interação com as métricas do sistema, acessando-se por `http://NO:19999`, sendo "NO" o endereço de *IP* da máquina na rede de forma descrita pela Sessão 2.1.2.1, além disso o

termo 19999 refere-se á porta de comunicação utilizada pela aplicação *netdata* ([WEB...](#), 2020b).

Figura 12 – Dashboard nativa do software netdata



Fonte: Elaborado pelo autor

Os menus referentes á seleção da métrica são mostrados no lado direito como pode ser observado na Figura 12, são geradas sessões para cada diferente classificação de métrica e várias sub sessões são criadas agrupando-se em família de métrica. Família é uma instancia que precisa ser tratada e exibida separadamente de instancias similares. Ora, diferentes instancias apresentam uma estruturação de dados em comum diferindo-se em quanto á classificação (DIKE, 2006). Por exemplo, sob o menu com classificação de métrica do tipo disco são agrupados sub sessões referentes á cada disco rígido bem como a cada prática reconhecida pelo *software*, de modo que a programação intrínseca do sistema se utiliza dessa hierarquia para plotar gráficos (WEB. . . , 2020b).

Na *dashboard* muitos dos gráficos plotados são acompanhados de breves descrições sobre o parâmetro que está sendo monitorado, contendo explicações conceituais facilitadoras no processo de detecção de erros. Os gráficos são gerados pelos coletores de métricas e são ferramentas de visualizações interativas do comportamento de uma métrica no sistema, define-se dimensão ou série quanto aos dados que estão sendo plotado graficamente, assim um gráfico com o nome *system.cpu* apresenta diferentes dimensões, uma vez que são tratados diferentes parâmetros no monitoramento da *cpu* (WEB. . . , 2020b).

Além disso, vários gráficos podem ser agrupados sob um contexto, sendo o contexto uma forma de agrupamento de gráficos baseados no tipo de métrica coletada e dimensões exibidas, como por exemplo a sessão de disco apresenta vários contextos como *disk.io*, *disk.backlog* etc, o *netdata* se utiliza da contextualização para criar gráficos individualmente e de forma posterior agrupá-los por família de métricas. de modo que a própria nomenclatura segue essa precedência (WEB. . . , 2020b). Ora, sendo *sdb* uma partição do sistema *disk_io.sdb* indica conjunto de gráficos relacionados ao disco para a partição, Por exemplo, ao mover o curso do mouse acima da lista de dimensões o *netdata* irá exibir o coletor que produz o gráfico e o seu contexto.

Graficamente são atribuídos á *dashboard* valores positivos, em verde, para métricas que estão executando leitura, bem como á estruturas de entrada do tipo *input* e *inbound*. Enquanto valores negativos são associados ao processo de escrita e de saída do tipo *output*, *outbound*, *sent* (WEB. . . , 2020b).

O *software netdata* é projetado para funcionar de forma distribuída, isto é, operando independentemente em cada nó do sistema, coletando e criando gráficas apenas para o sistema em que foi instalado (LEARN. . . , 2020). Dessa forma, as informações á respeito da *dashboard* são armazenadas sobre cada gráfico individualmente no arquivo *dashboardinfo.js* gerado com a instalação da aplicação, o arquivo inclui descrição, coloração, titulo entre outras informações necessárias para renderizar a *dashboard* e podem ser customizados (WEB. . . , 2020b). Utiliza-se desse arquivo diretamente por meio de qualquer página *HyperText Markup Language (HTML)* de modo a renderizar os gráficos.

No entanto, a aplicabilidade da *dashboard* não está limitada exclusivamente a

um nó, é possível por meio do conhecimento de redes promover a comunicação entre os nós se utilizando da ferramenta de *streaming* implementada no *software* e ainda manipular as métricas remotamente para gerar uma *dashboard* unificada para todos os nós ([STREAMING...](#), 2020a).

3 Materiais e métodos

Nesta seção será descrito de forma estrutural a instalação e configuração do *software* utilizado no *Cluster* do tipo *Beowulf* do laboratório LPS.

3.1 Materiais

Para implementação eficiente da ferramenta de análise de métricas utilizada chamada *netdata* foram utilizados conceitos básicos de funcionamento de um *Cluster*, ferramentas de configuração de rede a nível do sistema operacional *linux*, além de explorar extensivamente o protocolo *SSH*, responsável por estabelecer a conexão remota com os nós da rede e estruturado de modo a viabilizar manipulações e transferência de arquivos diretamente do terminal *linux* (NEGUS, 2020). Além disso a linguagem de programação *JavaScript* e *HTML* podem ser empregadas na criação e modificação da interface interativa *dashboard* implementada pelo *software netdata* (WEB..., 2020b).

Opta-se pelo *software netdata* para aquisição, processamento e exibição das métricas por ser uma ferramenta de código aberto, constantemente atualizada (LEARN..., 2020). Apresentando fator de escalabilidade por meio de diversas colaboração através de ferramentas com repositórios abertos no *GitHub*, envolvendo a comunidade de colaboradores e desenvolvedores globalmente (GITHUB...,). A ferramenta adéqua-se às necessidades de monitoramento da rede, por meio da exibição de métricas em tempo real e de forma customizada, utilizando-se de *dashboard* interativas e programáveis para visualização e monitoramento de todos os nós do sistema de *clustering* simultaneamente (WEB..., 2020b). Isto é, torna-se possível visualizar as métricas do laboratório LPS e detectar eventuais problemáticas sobre a execução de códigos implementados excedendo o uso das capacidades computacionais em determinado nó, e dessa forma a atender às necessidades dos pesquisadores quanto ao gerenciamento de recursos.

O sistema operacional *linux* foi escolhidos por ser *open sourcer*, gratuito e permitir configuração remota (ABBOTT, 2013). Além disso, *linux* possui os protocolos de redes integrados de forma a otimizar a comunicação e análise de métricas (NEGUS, 2020). A vantagem da estruturação do *linux* em código aberto permite maior eficiência e otimização no processo de configuração uma vez que existe *packages* disponível que auxiliam no processo de configuração e atualização de *softwares* (SANDERS, 2017).

As configurações foram realizadas em um *Cluster* do tipo *Beowulf* composto por 12 *CPU Dell* com processador *Interl(R) Core(TM) i7 – 4600U @2.10GHZ 2.70GHZ*, memória *RAM* de 16,0GB, sistema operacional sendo uma distribuição *linux* com *Fedora*

workstation e executado na arquitetura de 64 *bits*. O nó principal do *Cluster* utiliza uma distribuição do tipo *Fedora Server* o ambiente de trabalho é do tipo *GNOME desktop* e sua interface é *GNOME Shell* os outros nós da rede são acessados remotamente por meio do *SSH*.

3.2 Métodos

Nesta seção serão introduzidos os métodos para a implementação do *software netdata*, responsável pela aquisição e exibição das métricas do *Cluster*. A estrutura de exibição de métricas com usuário é modificada de modo a atender as necessidades do laboratório LPS. Todos arquivos de configuração desenvolvidos estão disponível no repositório remoto da ferramenta, disponível em: <<https://github.com/eduardoalmeidalps>> Entre estes, os *script* em *bash* para instalação e configuração da ferramenta, os arquivos de configuração dos nós (com extensão *.conf*) configurados para a comunicação e envio de métricas, bem como os códigos em *html* da *Dashboard* gerada exclusivamente para as métricas do laboratório LPS. As informações referentes á rede interna, como *MAC adress*, *IP* dos nós constituintes da *LAN*, *gateway* da rede e *IP* do servidor *SSH* foram ocultados de modo a preservar a integridade da rede do laboratório.

Para implementar a funcionalidade *multi dashboard* e objetivo do projeto, responsável por disponibilizar todas as métricas do sistema de *clustering* simultaneamente, deve-se abordar as seguintes etapas de configuração:

- Acesso ao *cluster*
- Mapeamento da rede
- Instalação dos pacotes de dependências
- Instalação do *software Netdata*
- Configuração do modo memória *dashboard*
- Implementação da ferramenta *streaming*
- Implementação da *dashboard*

3.2.1 Acesso ao *cluster*

Por meio de qualquer dispositivo, ou inclusive remotamente, é possível usar o protocolo *ssh* de modo á acessar o servidor *shell* (KIDWAI et al., 2020), cuja configuração o estrutura como nó principal do sistema de *cluster* (BARRETT DANIEL J.; SILVERMAN, 2009). Dessa forma, é possível iniciar a configuração diretamente presencialmente no

laboratório por meio da interface de conexão com usuário da própria máquina ou então por conexão remota de um dispositivo fora da *WLAN* do laboratório. No entanto, para acessar remotamente o *cluster* é necessário conhecer a priori o endereço real *IP* do servidor *SSH* (GROPP EWING LUSK, 2003).

Caso a inicialmente a rede já esteja implementada e a instalação do *cluster* estabelecida existem diferentes formas de se visualizar o *IP* por meio do terminal. Utiliza-se por exemplo, o `ip addr show` para se obter tanto o *IPv4* quanto o *IPv6* associados àquele dispositivo (ABBOTT, 2013).

Uma vez conhecido o *IP* com a devida autenticação por uma chave de acesso, configurada na instalação do servidor *ssh*, é possível acessar a máquina por meio do comando `ssh user@ip`, sendo *user* o nome de usuário atribuído na criação do servidor que se está tentando acessar, o campo *IP* é referente ao endereço *IPv4* "real" desse servidor, isto é, sem atribuição *DCHP* promovida pelo roteador para endereçamento na rede interna, de acordo formatação e características explicitada na Sessão 2.1.2.1 (BARRETT DANIEL J.; SILVERMAN, 2009).

A sintaxe de acesso *SSH* é essencial para configuração dos demais nós da rede e objetivando generalização de solução será utilizada também para o nó principal. Ora, apenas o nó principal possui dispositivos de entrada e saída de modo que seria possível configurá-lo por meio de *GUI*, no entanto os demais nós devem ser configurados remotamente.

3.2.2 Instalação *netdata*

3.2.2.1 Instalação dependências *netdata*

Uma vez com acesso ao diretório principal do *cluster* é então iniciada a etapa de instalação da ferramenta utilizada para objetivo do projeto. Primeiramente é necessário estar com *bash* funcional, de modo a instalar pelo terminal os pacote de dependências necessários para o funcionamento do *software netdata*. Os pacotes de dependência para instalação do *software* e funcionamento adequado de *plugins* podem ainda ser instalados *offline* diretamente por meio da execução do *script* `dependencia.sh` disponível no *Git* do projeto ¹.

Adicionalmente, é possível instalar online por meio do terminal com acesso ao repositório da distribuição pelo sistema operacional, como explicado na Sessão 2.2.8. Inicialmente, instala-se o comando `curl`, por meio do *script* disponibilizado na sessão de Apêndice 5 (INSTALLER. . . , 2020). Além disso, o *script* executa o comando apontando para o conteúdo da *url* onde estão os pacotes de dependências instalando-os no código com a linha seguinte por meio do comando `dnf install` os pacotes selecionados são instalados

¹Disponível em: <<https://github.com/eduardoalmeida1522/netdataips>>

ou atualizados como mostrados na Figura 13 e o *script* está disponível no repositório do projeto com o nome `curl.sh`.

Figura 13 – Pacotes de dependências *netdata*

Package	Arch	Version	Repository	Size
Instalando:				
Judy-devel	x86_64	1.0.5-21.fc31	fedora	71 k
autoconf	noarch	2.69-31.fc31	fedora	666 k
autoconf-archive	noarch	2019.01.06-4.fc31	fedora	589 k
autogen	x86_64	5.18.16-3.fc31	fedora	589 k
automake	noarch	1.16.1-13.fc31	fedora	666 k
git	x86_64	2.25.4-1.fc31	updates	125 k
libmnl-devel	x86_64	1.0.4-10.fc31	fedora	32 k
libuuid-devel	x86_64	2.34-4.fc31	updates	28 k
libuv-devel	x86_64	1:1.40.0-1.fc31	updates	29 k
lz4-devel	x86_64	1.9.1-1.fc31	fedora	28 k
openssl-devel	x86_64	1:1.1.1g-1.fc31	updates	2.2 M
Atualizando:				
openssl	x86_64	1:1.1.1g-1.fc31	updates	665 k
openssl-libs	x86_64	1:1.1.1g-1.fc31	updates	1.4 M
perl-Errno	x86_64	1.30-455.fc31	updates	24 k
perl-interpreter	x86_64	4:5.30.3-455.fc31	updates	6.1 M
perl-libs	x86_64	4:5.30.3-455.fc31	updates	1.7 M
Instalando dependências:				
Judy	x86_64	1.0.5-21.fc31	fedora	134 k
autogen-libopts	x86_64	5.18.16-3.fc31	fedora	75 k
git-core	x86_64	2.25.4-1.fc31	updates	4.8 M
git-core-doc	noarch	2.25.4-1.fc31	updates	2.2 M
guile	x86_64	5:2.0.14-17.fc31	fedora	3.4 M
libuv	x86_64	1:1.40.0-1.fc31	updates	151 k
m4	x86_64	1.4.18-11.fc31	fedora	217 k
perl-Error	noarch	1:0.17028-1.fc31	fedora	42 k
perl-Git	noarch	2.25.4-1.fc31	updates	44 k
perl-TermReadKey	x86_64	2.38-4.fc31	fedora	36 k
perl-Thread-Queue	noarch	3.13-439.fc31	fedora	22 k
Resumo da transação				
=====				
Instalar	22 Pacotes			
Atualizar	5 Pacotes			
Tamanho total do download: 26 M				

Fonte: Elaborado pelo autor

Instalam-se outras dependências que serão utilizadas nos diferentes processos de instalação disponibilizado pelo desenvolvedor, principalmente relacionadas à linguagem *c*, como por exemplo o compilador *GCC* e os pacotes responsáveis pela manipulação e execução desse tipo de arquivos pelo sistema operacional (HAGEN, 2006). Nota-se que o cluster é construído com uma distribuição *linux* do tipo *fedora* que utiliza pacotes do tipo *RPM* implementando o comando *yum* referente à manipulação de arquivos em *linux* (DOCS...,). Dessa forma, o comando *apt-get* extensivamente utilizado em distribuições baseadas em *debian* não está disponível nativamente (BRESNAHAN, 2020).

3.2.2.2 Instalação efetiva do *netdata*

Depois de instalado os pacotes de dependências requeridos para o funcionamento do *software* é utilizado o *script* em *bash* desenvolvido por meios da diretivas de instalação do próprio desenvolvedor (INSTALLER..., 2020). O código completo está disponível na sessão de Apêndice B. O *script* consiste em viabilizar o uso do comando de acesso a repositórios *git* de modo a acessar o conteúdo do repositório oficial do desenvolvedor do *software netdata* (GIT..., 2020). Assim, inicialmente são instaladas algumas dependências do comando *git* (GITHUB...,). Além disso, por meio dos comandos *cd*, *yum* e *install* o *script* copia o repositório em questão para o diretório *home/lps* do cluster bem como instalar

as ferramentas necessários para o funcionamento do *software netdata* nesta máquina. Como instrumento didático explicita-se na Figura 14 um *printscreen* referente ao *script* disponível no apêndice, de modo a facilitar o entendimento do código.

Figura 14 – *Script* de instalação *netdata*

```
1 # Download do NETDATA em FEDORA
2
3 # Atualizar dependencias do sistema
4 sudo yum update
5
6 # Dependencias adicionais Git
7 sudo yum install asciidoc xmlto docbook2X getopt
8
9 # Instalar comando Git
10 sudo yum install git-all
11
12 # Descarregar arquivos - pasta 'netdata' é criado em /home/lps
13 git clone https://github.com/netdata/netdata.git --depth=100
14 cd netdata
15
16 # Executar script em /home/lps/netdata-installer.sh
17 sudo ./netdata-installer.sh
18
```

Fonte: *Print* do *script* de instalação disponível no Apêndice 5

3.2.2.3 Comandos para gerenciamento *netdata*

Após instalado o *software*, é importante salientar que alguns comandos são essenciais para o gerenciamento da ferramenta, sejam para modificação ou atualização da aplicação (INSTALLER..., 2020). Como por exemplo, os comandos utilizados para iniciar e pausar uma aplicação do *netdata*, respectivamente `sudo systemctl enable netdata` e `sudo systemctl stop netdata` (MASTER..., 2020). Ora, tal sintaxe é utilizado nas distribuições *linuxs* baseada *Systemd* cuja aplicabilidade é referente ao gerenciamento de serviços e sistema e é está intrinsecamente relaciona á inicialização de aplicações e do próprio sistema operacional (*boot*). *Systemd* é executado de forma continua no sistema desde a sua inicialização, é definido como pertencente á uma classe de processo chamado *daemon*, inclusive responsável pelo gerenciamento de outras aplicações dessa classe, sendo o primeiro processo a ser inicializado pelo sistema operacional (NEGUS, 2020). Dessa forma, é interessante adicionar a aplicação *netdata* á inicialização do sistema, assim caso o *cluster* seja reiniciado por algum motivo não será necessário executar em cada nó novamente o comando `sudo systemctl enable netdata`, o *script* comentado está na sessão de Apêndice C. A criação de *scripts* mesmo com códigos curtos justifica-se na implementação no projeto pois é possível enviar para os demais nós do *cluster* o arquivo *.sh* e executa-los rapidamente. A transferência se dá por uma implementação do *ssh* chamada de *STFP*, *secure transfer protocol*, com sintaxe da forma: `scp ini.sh usuario@usuario2`, sendo usuário e usuário2 a nomenclatura atribuída na criação do servidor *ssh* ao nó principal e o nó de destino do arquivo (BARRETT DANIEL J.; SILVERMAN, 2009).

Entre os principais comandos é importante citar o comando `sudo systemctl list --units --type=service` para verificar todos os serviços que estão sendo executados naquele sistema

operacional e gerenciar os recursos (NETDATA..., 2020a). Além disso, é recomendável reiniciar a aplicação do *netdata* por meio do comando `sudo service netdata restart` após cada modificação realizada em seus arquivos de configuração de modo a se certificar de que as alterações serão empregadas (INSTALLER..., 2020).

Para eventual atualização do *software* basta localizar o diretório criado pelo *script* desenvolvido nessa sessão, isto é: `/home/lps/netda/`. De modo a obter a versão mais recente do *software* a partir do repositório oficial, é utilizado o comando `git pull` e assim, efetivamente instalar o *script* baixado por meio da execução do *script shell*: `sudo ./netdata-installer.sh`. Além disso, nesse mesmo diretório existe um *script* destinado à desinstalação do *software* de modo que de forma análoga basta executar a linha de código `sudo ./netdata-uninstaller.sh --force` para efetuar a desinstalação da ferramenta (INSTALLER..., 2020).

3.2.2.4 Diretórios *netdata*

Deve-se considerar a localização do diretório quando da manipulação e configuração de arquivos *netdata*. Ora, o diretório dos arquivos de configuração não coincide necessariamente com o diretório criado para instalação do *software* a partir do diretório *git*. Lembrando-se que é possível utilizar-se do comando *pwd* para exibir o diretório atual e alternar entre diretórios por meio *cd*.

Nota-se que devido à estruturação do servidor *ssh* o acesso ao *cluster* irá direcionar o usuário ao diretório: `/home/lps/`. Desse modo é necessário utilizar duas vezes o comando `cd ..` de modo a retroceder ao diretório raiz do sistema, isto é, ao utilizar o comando *pwd* em um diretório e o terminal retornar `/` indicando que esse é o diretório principal do sistema. A partir do diretório principal é possível localizar os subdiretórios criados pelo *netdata* durante o processo de instalação, como mostrado na Figura 15. Logo, os diretórios dos arquivos de configurações são acessados a partir de tal diretório por `cd etc/netdata` para os arquivos de extensão `.conf` e por `cd usr/share/netdata/web` para os arquivos de extensão `.html`

Nota-se que por ser uma implementação recente e depender de pacotes de terceiros cuja atualização não pode ser garantida pelo desenvolvedor *netdata*, esse método de instalação pode não ser compatível com todos sistemas operacionais, sendo necessário assim efetuar a instalação manual ([KICKSTART...](#), 2020) .

3.2.2.6 Conclusão do processo instalação

Uma vez que o *netdata* é efetivamente instalado no sistema é produzida no terminal a saída mostrada na Figura 16.

Figura 16 – Saída do instalador *netdata*

```
netdata by default listens on all IPs on port 19999,
so you can access it with:

http://this.machine.ip:19999/

To stop netdata run:

systemctl stop netdata

To start netdata run:

systemctl start netdata

Uninstall script copied to: /usr/libexec/netdata/netdata-uninstaller.sh

--- Installing (but not enabling) the netdata updater tool ---
Failed to disable unit: Unit file netdata-updater.timer does not exist.
Update script is located at /usr/libexec/netdata/netdata-updater.sh

--- Check if we must enable/disable the netdata updater tool ---
You chose "NOT" to enable auto-update, removing any links to the updater from cron (it may have happened if you are reinstalling)

--- Wrap up environment set up ---
Preparing .environment file
[/home/lps/netdata]# chmod 664 /etc/netdata/.environment
OK

Setting netdata.tarball.checksum to 'new_installation'

--- We are done! ---

A
|.....netdata
|.....is installed and running now!
|.....
+-----+
enjoy real-time performance and health monitoring...
```

Fonte: Elaborado pelo autor

Com a instalação devidamente concluída no nó principal da rede é necessário ainda instalar o *software netdata* em todos os nós com quais se deseja comunicar e obter métricas, dessa forma são repetidas as instruções dessa sessão em todas as máquinas constituintes da rede por meio do *ssh* explicitado na Sessão 2.2.5. A instalação segue a mesma estruturação, sendo necessário inciar a sessão no dispositivo a partir de uma conexão pré estabelecida com o servidor principal de modo que a partir do nó principal utiliza-se da sintaxe: *ssh usuario@usuario2*, sendo usuário2 o nó em questão ([BARRETT DANIEL J.;SILVERMAN, 2009](#)).

3.2.3 Configuração *netdata*

Os computadores constituintes do *cluster* são conectados por meio de *switch* que permite a comunicação na *LAN*, a conexão entre a *NIC* dos computadores e o *switch* se dá por meio de cabeamento RJ45 utilizando o protocolo de comunicação do tipo *ethernet*.

Dessa foma, é necessário configurar a rede e integrar o *NFS* para os nós de computação, . O servidor *NFS* é atribuído ao primeiro nó que passa a funcionar como o nó principal ([GROPP EWING LUSK, 2003](#)).

As configurações de rede implementadas seguem a mesma lógica e estrutura de comunicação estabelecida pelo protocolo *TCPIP* e com endereçamento por *IPv4* descritos na Sessão 2.1.2.1. *Clusters* do tipo *Beowulf* normalmente utilizam a faixa de endereços especiais *IPv4* reservados para *LAN* (GROPP EWING LUSK, 2003), de acordo com a Tabela 2, estes endereços são permanentemente não atribuídos, isto é, não são encaminhados por *routers* constituintes no *backbone* da *internet*, logo não entram em conflito com endereços *IP* publicamente endereçáveis (WHITE, 2018).

O objetivo dessa sessão é estabelecer a comunicação entre os nós do *cluster* por meio dos comandos de rede em *linux*. Uma vez que cada aplicação instalada do *netdata* é capaz de replicar a base de dados interna para outra aplicação *netdata* por meio da rede interna e essa comunicação se dá por meio da ferramenta de *streaming* incluída pelo próprio desenvolvedor como explicado na Sessão 2.4.1 (STREAMING..., 2020a). Sendo assim, tal ferramenta é dependente dos parâmetros e topologia da rede interna, sendo necessário configurar o endereçamento de *IP* e portas explicados na Sessão 2.1.2.1 para o sistema operacional *linux* (GROPP EWING LUSK, 2003). Ora, são os parâmetros de rede que serão efetivamente manipulados pelo *software netdata* permitindo a comunicação e possibilitando a funcionalidade da ferramenta de *streaming* e portanto essa ferramenta precisa ser configurada para cada rede particular de forma distinta.

3.2.3.1 Mapeamento da rede

Com o mapeamento da rede é possível saber os parâmetros envolvidos para a configuração da ferramenta de *streaming* para a rede especificada (ELTRINOS, 2020). Uma vez que diversas topologias podem ser empregadas e diferente números de *IP* atribuídos dependendo do fabricante dos dispositivos. Ao digitar os comandos de gerenciamento de rede detalhados com o *route* o sistema operacional irá informar a tabela de roteamento do *kernel* do *linux* e fornecer um panorama geral da topologia da rede, além disso para mais detalhes sobre a interface basta digitar `ifconfig -a`, a saída gerada pelos comandos é mostrada na Figura 17 (BOTH, 2020).

Figura 17 – Detalhamento interface de rede

```

Last login: Thu Nov 12 16:25:24 2020 from 201.21.142.228
[~]$ ifconfig -a
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1. netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::e269:95ff:fe68: prefixlen 64 scopeid 0x20<link>
    ether e0:69:95:68 9c txqueuelen 1000 (Ethernet)
    RX packets 302144444 bytes 266216463923 (247.9 GiB)
    RX errors 0 dropped 24 overruns 0 frame 0
    TX packets 185047852 bytes 16012698939 (14.9 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 20 memory 0xf7400000-f7420000

enp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 50 netmask 255.255.255.0 broadcast 5.255
    ether 00:e0:53:14:2 txqueuelen 1000 (Ethernet)
    RX packets 203180073 bytes 14895931436 (13.8 GiB)
    RX errors 0 dropped 143898 overruns 0 frame 0
    TX packets 7941471 bytes 2308221783 (2.1 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Loopback Local)
    RX packets 194311 bytes 114329727 (109.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 194311 bytes 114329727 (109.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Fonte: Elaborado pelo autor

Dessa forma, na exibição de rotas mostrada na Figura 17 percebe-se de fato a existência de duas redes, ambas capazes de endereçar 252 dispositivos, uma vez que estão sob uma máscara de sub-rede da forma 255.255.255.0 de acordo com a Sessão 2.1.2.1. O endereço da rede eno3 é atribuído ao servidor *ssh* enquanto o endereço da rede eno1 192.168.1.x representa uma faixa reservada e é comercialmente atribuído pelo roteador (D.E., 2000). Observa-se que o endereço da rede do servidor *ssh* foi omitido na Figura 17 uma vez que o comando retorna o *IP* real e acessível do servidor.

Dessa forma, tem-se duas *NIC* associadas cujo *MAC* é explicitado pelo comando `ifconfig -a`. Dessa forma, o servidor *ssh* sendo associado a interface *enp3s0* por meio de um *switch* e a outra interface faz a conexão com a rede de *internet* por roteador diretamente a interface de rede *eno1* (PERLMAN, 1999). Dessa forma, devido a presença do roteador *wireless* e considerando todas condições referentes à conectividade da Sessão 2.1.1 é estabelecida uma rede *WLAN* interligando às redes e incluindo diretamente os nós do *cluster* (ALPERN; SHIMONSKI, 2010).

O endereço de *IP* interno associado à rede roteada conectada à *NIC eno1* é mostrado sob a forma 192.168.1.Y, de modo que é referenciada à sintaxe dos possíveis nós dessa rede os endereços no intervalo 192.168.1.2 - 192.168.1.254. Ora, três endereços são reservados a priori: o endereço do *host* 192.168.1.0, o *gateway* 192.168.1.1, bem como o endereço de *broadcast* dessa que é rede é 192.168.1.255 não podendo estes serem endereçado por nenhum computador (Nguyen et al., 2009).

Normalmente a configuração de *IP* estático já é atribuída na construção do *cluster* de modo a otimizar o seu desempenho (GROPP EWING LUSK, 2003). Dessa forma, seria suficiente mapeá-la para a implementação de aplicações que se utilizarão dos parâmetros associados à rede para implementação. Sendo possível atribuir ou alterar essas configurações

por meio do terminal com a configuração de super usuário `—su`, acessando-se o utilitário de configuração e atribuindo-se manualmente endereço de *IP*. A configuração se dá por associação direta com endereço *IP* ao nome da interface de rede atribuído pelo sistema operacional (SHOTTS, 2019). Por exemplo, alterando a configuração de endereçamento lógico estático referente à *NIC* referenciada como *eno1* com: `sudo ip addr add [ip_atribuido] dev [eno1]`, sendo *ip_atribuido* um endereço de *IPv4* como citado na Sessão 2.1.2.1 (SOBELL, 2013).

Também é possível alterar ou atribuir um *IP* estático mudando os valores diretamente no arquivo de configuração do próprio sistema operacional, para tal utiliza-se o editor de arquivos `vim /etc/sysconfig/network-scripts/ifcfg-eno1`, possibilitando-se assim a atribuição de valores diretamente de acordo a especificidade do projeto (ELTRINOS, 2020). Por exemplo, alterando-se o valor do campo *IPADDR* no arquivo de configuração para *IPADDR=192.168.1.X*, sendo *x* números natural entre 3-253. De modo a exemplificar as configurações do projeto sem expor a topologia de rede do laboratório adota-se que o *IP* estático para o nó principal na rede local da forma *IPADDR = A.B.C.D* sendo esse um *IP* classe C referido na Tabela 2 O projeto ainda se referencia como *IP* real do servidor *SSH* como da forma *IPADDR = X.Y.Z.W* (PETERSON, 2003).

Além disso, é interessante também mapear as portas de comunicação que estão habilitadas no sistema. Ora, assim o protocolo *SSH* atua se utilizando da porta 22, como toda aplicação ao *netdata* é atribuída uma porta de comunicação, no caso 19999 para garantir a comunicação e possibilitar a análise de métricas (INSTALLER..., 2020).

Para tal utiliza-se a ferramenta de código aberto *nmap*³ utilizada para mapeamento de redes e implementada nativamente no repositório de *softwares* da maior parte das distribuições *linux*. Repositório esses que são acessados por meio de `dnf` para distribuições baseadas em pacotes do tipo *RPM* ou por `apt-get` em distribuições *debian*. Todos os pacotes presente nos repositórios oficiais são livres, de acordo com a definição de código aberto sendo disponibilizados pelas própria distribuição *Fedora*⁴. Bem como são listado os pacotes para distribuições baseadas em *debian*⁵ (DOCS...,).

Dessa forma, é facilmente instalada por meio do terminal diversos pacotes de dependência, instala-se o *nmap* por meio do comando `sudo dnf install nmap` (NMAP..., 2020). Na Figura 18 é mostrada o mapeamento feito para as portas de aplicação no nó principal do *cluster* `nmap -v -p1-20000 A.B.C.D`, sendo *A.B.C.D* o *IP* estático local. Observa-se ainda informações referentes à porta de aplicações conhecidas, como por exemplo a porta 22 utilizada para comunicação por *SSH*, ou então porta 80 referente à servidor *web* (NMAP..., 2020). As aplicações cujas portas são reservadas pelo sistema

³Disponível em: <<https://nmap.org/>>

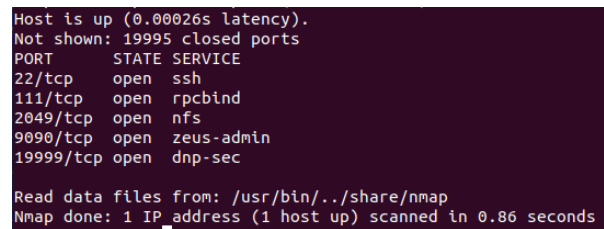
⁴Disponível em: <https://fedora-archive.ip-connect.info/fedora/linux/releases/29/Everything/x86_64/os/>

⁵Disponível em: <<https://www.debian.org/distrib/packages>>

podem ser obtidas por meio do comando *cat* explicitando como parâmetro o diretório de informação sobre portas no sistema operacional: `cat /proc/sys/net/ipv4/ip_local_reserved_ports` (MACH, 2020).

Nota-se ainda que a além do mapeamento de portas a ferramenta *nmap* possui diversas aplicabilidades do gerenciamento de uma rede, sendo instalada nativamente em algumas distribuições *linux*. Dessa forma, para obtenção de mais parâmetros basta digitar `nmap -h` e utilizar os comandos associados às informações desejadas (NMAP..., 2020).

Figura 18 – Mapeamento de portas



```
Host is up (0.00026s latency).
Not shown: 19995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp    open  rpcbind
2049/tcp   open  nfs
9090/tcp   open  zeus-admin
19999/tcp  open  dnp-sec

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.86 seconds
```

Fonte: Elaborado pelo autor

Muitos dos problemas decorrentes na implementação do *netdata* se dão por meio da restrição do *firewall* do sistema operacional quanto à porta de aplicação (DEAL, 2004). Logo, habilita-se a porta usada pela aplicação a priori por meio dos comandos `sudo firewall-cmd --add-port=19999/tcp`, tornando-se a alteração permanente para o sistema operacional por meio de `sudo firewall-cmd --runtime-to-permanent` e finalmente reiniciando a aplicação com `sudo firewall-cmd --reload` (INSTALLER..., 2020).

3.2.3.2 Configuração *streaming*

Na Sessão 2.4.1 foi descrito qualitativamente o conceito de *streaming* no *software netdata*, bem como definido suas especificidades e nomenclatura, esta sessão tem como objetivo configurar essa ferramenta da aplicação de modo a estabelecer propriamente a conexão entre todos os nó do sistema de *clustering*.

O nó principal do sistema de *clustering* será configurado de modo a receber as métricas da rede, sendo denominado dessa forma como *parent node* do sistema, enquanto os demais nós serão configurados de modo a enviar métricas e portanto são chamados de *child nodes*, de acordo com a classificação da Sessão 2.4.1 referente à ferramenta de *streaming* (STREAMING..., 2020a). Logo, como inicialmente foram atribuídos endereços *IPv4* fixos para cada dispositivo da rede *LAN* é possível a distinção de cada nó bem como o seu referenciamento diretamente na topologia de rede, viabilizando a configuração dos dispositivos na disposição de *streaming* projetada.

Para configurar a ferramenta de *streaming* inicialmente é preciso configurar o nó principal como *parent* e os outros nós do sistema de *clustering* como *child* de acordo com a

classificação estabelecida (STREAMING... , 2020a). Para tal, utilizam-se os comandos de gerenciamento, manipulação e edição de arquivos em *linux*. Como a configuração deve ser realizada em todos os nós, e como apenas o nó principal possui ambiente gráfico e dispositivos de entrada e saída associado então as configurações em todos demais nós devem ser feitas remotamente por meio do *SSH* no terminal (BARRETT DANIEL J.; SILVERMAN, 2009).

Dessa forma, com o *IP* estático e com a chave *API* gerada na Sessão 2.4.2 é possível estabelecer de forma segura a comunicação entre os nós em que estão instalados o *software* (GAO et al., 2021). Configuram-se os nós de modo que armazenem separadamente a suas respectivas base de dados e enviem as métricas para o nó principal para a criação do painel de métricas unificadas, segundo a disposição da Tabela 3.

Com a instalação da aplicação *netdata* descrita na Sessão 3.2.2 é criado no sistema operacional entre outros arquivos o arquivos de configuração global da aplicação principal *daemon* e o arquivo de configuração de *streaming*, ambos com extensão *.conf* e localizadas nos diretórios */etc/netdata*, os arquivos são a cada instalação do *software* em cada nó e devem ser individualmente configurados (STREAMING... , 2020a).

Nota-se que ao reinstalar a aplicação do *netdata* mais recente até a presente data pela implementação *kickstart* o arquivo *stream.conf* não foi criado no repositório referente á configuração. Dessa forma, este foi copiado do diretório replicado a partir do *Git* no processo de instalação, disponível em: *home/lps/netdata/streaming*. Dessa forma, foi necessário copiar manualmente o arquivo *stream.conf* desse diretório diretamente para o diretório de configuração reconhecido pelo *netdata*, para tal utiliza-se o comando *mv sudo mv stream.conf ///etc/netdata/stream.conf*. Logo, com o arquivo *stream.conf* devidamente localizado no diretório *///etc/netdata/stream.conf* o programa conseguirá reconhecer alterações promovidas no arquivo de configuração e promover alterações, enfatiza-se que sempre que algum parâmetro do arquivo de configuração for alterado é necessário reiniciar a aplicação *netdata* como descrito na Sessão 3.2.2 sobre instalação (MASTER... , 2020).

Assim, por meio do comando *cd* é possível acessar esse diretório e manipular esses arquivos de configuração com o editor de arquivo de preferencia, pode-se utilizar o editor *vim*, *nano* ou então o próprio *script* de configuração *script-config* incluído com o *netdata* (INSTALLER... , 2020). Nota-se que nos arquivos de configuração assim como nos arquivos *.sh* do *linux* o símbolo *#* indica comentário, de modo que muitas linhas de configurações nos arquivos *.conf* estão comentadas, logo ao modificar os parâmetros referentes á atribuição é importante deletar o símbolo *#* para a atribuição entrar em vigor, caso contrário o programa continuará a carregar as configurações padrões para aqueles parâmetros. Os comentários descrevem qualitativamente os parâmetros associados e muitas vezes emprega exemplos de modo a elucidar as configurações (LIB... , 2020).

No mesmo diretório dos arquivos de instalação o desenvolvedor disponibilizar um

script em *bash* intitulado *edit-config* de modo que nos tutoriais e guias⁶ disponibilizados pelos colaboradores e desenvolvedor se utilizam constantemente da sintaxe `sudo ./edit-config arquivo.conf` para a edição de arquivos de extensão *.conf* (STREAMING..., 2020a). Ao executar esse arquivo diretamente por meio do terminal é possível analisar o código fonte do *script* bem como a sintaxe utilizada detalhada sobre o seu uso, é especificado também o diretório de todos arquivos de configurações editáveis por esse *script* criado pelo *netdata*. Editando-se o arquivo de instalação é necessário reiniciar a aplicação *netdata* (MASTER..., 2020). O *script* se utiliza do ambiente do editor de arquivo do sistema operacional, isto é, editor *nano* ou *vim* na maioria dos casos (EDIT..., 2020).

3.2.3.2.1 Configuração *streaming* no nó principal

Para implementar a ferramenta de *streaming* alteram-se diretamente alguns parâmetros no arquivo de configuração principal do *netdata* em `/etc/netdata/netdata.conf` (INSTALLER..., 2020). Como mostrado na Tabela 3 para possibilitar a criação de uma *dashboard* interativa os campos no arquivo de configuração referente á operação *web* e modo de memória devem ser habilitados, nativamente eles estão com a configuração *none* (EDIT..., 2020).

Logo, deve-se acessar o diretório de configuração por meio de `cd etc/netdata` a partir do diretório raiz `/` e então inicializar o arquivo de configuração com o editor escolhido, isto é, caso opte-se pelo uso do *script* disponibilizado pelo desenvolvedor como descrito na sessão anterior, o comando será da forma: `sudo edit-config netdata.conf` (EDIT..., 2020). Ou então diretamente pelo editor do sistema operacional por: `sudo vim netdata.conf` (ABBOTT, 2013). Além disso, ressalta-se que no editor *vim* para habilitar qualquer modificação no arquivo utiliza-se a tecla *insert*, possibilitando então alterar os valores com o editor. Após a alteração basta digitar `:wq` para salva-las e sair do editor de texto (NEGUS, 2020).

Para a especificidade do projeto estruturação desejada para o nó principal deve ser capaz de armazenar as métricas e recebe-las por meio da rede interna configurada com *IP* estático. Na página da ferramenta sobre as configurações do *daemon netdata* estão descritas todas as sessões presentes no arquivo de configuração bem como suas funcionalidades⁷. Além disso, é possível visualizar diretamente as configurações atribuídas á determinada aplicação de *netdata* por meio do *browser* com `http://netdata.server.hostname:19999/netdata.conf`, sendo *netdata.server.hostname* o *IP* estático atribuído localmente na rede para aquele nó cujas configurações serão visualizadas. O *IP* estático foi configurado ou mapeado e atribuído de forma genérica no projeto como A.B.C.D para o nó principal. É possível ainda visualizar a disposição do arquivo de configuração em sua disposição padrão com uma exemplificação fornecida pelo desenvolvedor.⁸

⁶Disponível em: <<https://learn.netdata.cloud/guides>>

⁷Disponível em <<https://learn.netdata.cloud/docs/agent/daemon/config>>

⁸Disponível em: <<https://netdata.firehol.org/netdata.conf>>

Dessa forma, são atribuídos os seguinte modos de operação para o nó principal referentes aos parâmetros de memória e *web* no arquivo de configuração geral: [global].memory mode = Dbengine e [web].mode = static-threaded, isto é, no campo do código referente às configurações globais altera-se a linha de atribuição de memory mode = none para memory mode = Dbengine de acordo com as especificações para funcionalidade de nó principal referente na Tabela 3. Desse modo, devido às necessidades de implementação especificadas por meio de necessidade de armazenamento otimizado de métricas com menor custo computacional justifica-se a implementação pelo modo de memória *Dbengine* explicitado na sessão lrefmemodb (DBENGINE, 2020), Enquanto que a atribuição ao [web].mode = static-threaded se justifica pela Tabela 3 uma vez que a única disposição cuja atribuição seria nula seria é para uma topologia de estritamente coletora de métricas, ou *headless collector* de acordo com a Sessão 2.4.1 (STREAMING..., 2020a).

Os outros parâmetros para configuração se encontram no arquivo referente à configuração de *streaming* e são acessados e editados de forma análoga ao arquivo anterior, encontrando-se no diretório */etc/netdata*. Assim, o comando para habilitar a edição no arquivo de configuração analogamente será: `sudo vim stream.conf`.

Ao abrir o arquivo de configuração será exibida uma janela semelhante à mostrada na Figura 19, possibilitando-se então alterar os valores com o editor, a edição se dá de forma análoga à configuração anterior. Inicialmente, no início do documento é atribuído ao campo [stream].enabled = no (STREAMING..., 2020a). Ora, trata-se do nó principal cuja funcionalidade não é enviar seus dados por meio de *streaming* para um nó subsequente mas sim receber os dados e reproduzi-los em uma *dashboard*. Caso fosse atribuído *yes* a esse campo o nó passaria a atuar como um *proxie*, como definido na Sessão 2.4.1 e mostrado na Tabela 3.

Ainda nesse arquivo de configuração, ativa-se inicialmente a comunicação por chave *API* ao atribuir no arquivo a chave *API* gerada na Sessão 2.4.2 no campo [my-api-key], isto é, substituindo-se o texto *my-api-key* entre chaves pelo valor do código da chave *API* gerado com o *uuidgen* da Sessão 2.4.2, como mostrado na Figura 19.

Figura 19 – Alteração no campo *API* do *stream.conf*

```

# -----
# 2. ON PARENT NETDATA - THE ONE THAT WILL BE RECEIVING METRICS
#
# You can have one API key per child,
# or the same API key for all child nodes.
#
# netdata searches for options in this order:
#
# a) parent netdata settings (netdata.conf)
# b) [stream] section (above)
# c) [API_KEY] section (below, settings for the API key)
# d) [MACHINE_GUID] section (below, settings for each machine)
#
# You can combine the above (the more specific setting will be used).
#
# API key authentication
# If the key is not listed here, it will not be able to push metrics.
#
# [API_KEY] is [YOUR-API-KEY], i.e [11111111-2222-3333-4444-555555555555]
# [2xxxxx-xxxxx-11ex-aex-0xxxxx2]
# Default settings for this API key
#
# You can disable the API key, by setting this to: no
# The default (for unknown API keys) is: no
enabled = yes
#
# A list of simple patterns matching the IPs of the servers that
# will be pushing metrics using this API key.
# The metrics are received via the API port, so the same IPs
# should also be matched at netdata.conf [web].allow connections from
allow from = *
#
# The default history in entries, for all hosts using this API key.
# You can also set it per host below.
# If you don't set it here, the history size of the central netdata
# will be used.
default history = 3600
#
# The default memory mode to be used for all hosts using this API key.
# You can also set it per host below.
# If you don't set it here, the memory mode of netdata.conf will be used.
# Valid modes:
-- INSCRIÇÃO --

```

Fonte: Elaborado pelo autor

Ainda no campo referente à *API* na linha `[allow from = *` é utilizado o símbolo referente à habilitação de *wild card* sobre acessibilidade no *linux*. Ora, o *netdata* também suporta a padronização simples, isto é, a indexação do símbolo `*` de atribuição e generalização na busca de parâmetros (SIMPLE..., 2020). Dessa forma, a linha de configuração `allow from = *` significa que o nó principal está configurado de modo a receber dados de qualquer endereço de *IP* na rede interna, como mostrado Figura 19.

De forma adicional, quando existe necessidade de implementações de topologias mais complexas o *netdata* permite maior seletibilidade na disposição dos nós e tratamento dos parâmetros por meio da implementação do parâmetro *Machine GUID* citado na Sessão 2.4.1. Trata-se de um parâmetro adicional quando existe necessidade de atribuição de roteamento para as métricas, isto é, quando existem *proxy* e servidores envolvidos na topologia (C., 2015). O parâmetro pode ser explicitado para cada aplicação por meio do comando `sudo cat /opt/netdata/var/lib/netdata/registry/netdata.public.unique.id` no terminal da máquina cujo parâmetro deseja-se conhecer. Dessa forma, utiliza-se do valor obtido como saída no terminal dos nós secundários substituindo-o no campo `[MACHINE_GUID]` entre chaves presente no arquivo de configuração de *streaming* do *proxy* que receberá as métricas daquele nó (STREAMING..., 2020a). Obtendo-se disposição mostrado na Figura 20, é importante ainda nesta sessão habilitar este *host* e por conseguinte seu uso por meio do campo `enabled = yes` (STREAMING..., 2020a).

Figura 20 – Alteração no campo *MGUID* do *stream.conf* nó principal

```

# -----
# 3. PER SENDING HOST SETTINGS, ON PARENT NETDATA
#   THIS IS OPTIONAL - YOU DON'T HAVE TO CONFIGURE IT

# This section exists to give you finer control of the parent settings for each
# child host, when the same API key is used by many netdata child nodes / proxies.
#
# Each netdata has a unique GUID - generated the first time netdata starts.
# You can find it at /var/lib/netdata/registry/netdata.public.unique.id
# (at the child).
#
# The host sending data will have one. If the host is not ephemeral,
# you can give settings for each sending host here.

[2xxxxxx7a-xxxx8e-xxxxf-0xxxxx]
# enable this host: yes | no
# When disabled, the parent will not receive metrics for this host.
# THIS IS NOT A SECURITY MECHANISM - AN ATTACKER CAN SET ANY OTHER GUID.
# Use only the API key for security.
enabled = yes

# A list of simple patterns matching the IPs of the servers that
# will be pushing metrics using this MACHINE GUID.
# The metrics are received via the API port, so the same IPs
# should also be matched at netdata.conf [web].allow connections from
# and at stream.conf [API_KEY].allow from
allow from = *

# The number of entries in the database
history = 3600

# The memory mode of the database: save | map | ram | none | dbengine
memory mode = save

# Health / alarms control: yes | no | auto
health enabled = yes

# postpone alarms when the sender connects
postpone alarms on connect seconds = 60

# need to route metrics differently?
# the defaults are the ones at the [API KEY] section
#proxy enabled = yes | no
#proxy destination = IP:PORT IP:PORT ...
#proxy api key = API_KEY
#proxy send charts matching = *
-- INSCRIÇÃO --

```

Fonte: Elaborado pelo autor

Nota-se ainda que vários coletores podem ser direcionados á um mesmo *proxy* bastando para tal explicitar seu valor registrado *MGUID* em `/var/lib/netdata/registry/netdata.public.unique.id` no sistema operacional do nó secundário e adicionar outra sessão atribuindo-o com a sintaxe mostrada no Apêndice D, sob a sessão `[MACHINE_GUID]` nos arquivos de configuração de *streaming* do *proxy* que irá receber as métricas (STREAMING..., 2020b). Ora, pode-se assim direcionar diretamente as métricas geradas por meio de qualquer topologia desejada. Os valores *API* na Figura 19 e *MGUID* na Figura 20 não representam valores reais e forma alterados para manutenção de confiabilidade. Os arquivo mostrando os comparativos das configurações nativas e alteradas para as sessões de interesse dos arquivos *.conf* referente ao nó principal estão no Apêndice E. Devido á extensão do arquivo de configuração os parâmetros não alterados são omitidos no arquivo.

3.2.3.2.2 Nós secundários - *child*

Inicialmente, deve-se iniciar a sessão no dispositivo que deseja configurar como nó secundário. Uma vez que a configuração deve ser realizada em todos os nós secundários, e apenas o nó principal possui ambiente gráfico e dispositivos de entrada e saída associado então todas as configurações devem ser feitas por meio do *ssh* no terminal (NEGUS, 2020).

Dessa forma, uma vez estando conectado ao nó principal o acesso aos nós secundário se dá a partir de outra sessão *ssh* por meio do comando `ssh usuario@usuario2`, sendo `usuario2` o nó em questão (BARRETT DANIEL J.; SILVERMAN, 2009).

A configuração segue a mesma estruturação da sessão anterior, diferindo-se quanto aos valores de atribuição às variáveis associadas e alteração de alguns parâmetros adicionais. Assim, com a instalação da aplicação *netdata* descrita na Sessão 3.2.2 são criado no sistema operacional do nó em questão os arquivos *.conf* de configuração geral da aplicação principal e o arquivo para configuração de *streaming* `/etc/netdata/netdata.conf` e `/etc/netdata/stream.conf` respectivamente (STREAMING..., 2020a).

Para possibilitar a implementação de uma *dashboard* interativa de acordo com a Tabela 3 os campos no arquivo de configuração referente á operação *web* e modo de memória também devem ser habilitados, nativamente eles estão com a configuração *none*. Ora, para estruturação desejada os nó secundários devem ser capaz de armazenar as métricas, de modo que seja possível mensura-las e avalia-as individualmente, além de envia-las por meio da rede interna configurada com *IP* estático para o nó principal, possibilitando assim a reprodução comparativa de todas as métricas em uma única *dashboard* sendo executada pelo nó principal do *cluster*. Dessa forma, nesse campo as atribuições são as mesmas da sessão anterior: `[global].memory mode = dbengine` e `[web].mode = static-threaded` (STREAMING..., 2020a).

Na configuração referente ao *streaming*, acessado em `/etc/netdata` novamente com `sudo vim stream.conf` a mesma janela de configuração da sessão anterior será mostrada possibilitando então alterar os valores com o editor, a edição se dá de forma análoga á configuração anterior mas com a manipulação de diferentes parâmetros (STREAMING..., 2020b). Nota-se que nesta etapa de configuração as modificações estão restritas ao campo *1 On slave netdata - the one that will be sending metric* do arquivo de configuração como mostrado na Figura 21 (STREAMING..., 2020a).

Figura 21 – Alteração no nó secundário do *stream.conf*

```
# netdata configuration for aggregating data from remote hosts
#
# API keys authorize a pair of sending-receiving netdata servers.
# Once their communication is authorized, they can exchange metrics for any
# number of hosts.
#
# You can generate API keys, with the linux command: uuidgen
#
# -----
# 1. ON CHILD NETDATA - THE ONE THAT WILL BE SENDING METRICS
[stream]
# Enable this on child nodes, to have them send metrics.
enabled = yes

# Where is the receiving netdata?
# A space separated list of:
#
# [PROTOCOL:]HOST[%INTERFACE][:PORT][:SSL]
#
# If many are given, the first available will get the metrics.
#
# PROTOCOL = tcp, udp, or unix (only tcp and unix are supported by parent nodes)
# HOST      = an IPv4, IPv6 IP, or a hostname, or a unix domain socket path.
#             IPv6 IPs should be given with brackets [ip:address]
# INTERFACE = the network interface to use (only for IPv6)
# PORT      = the port number or service name (/etc/services)
# SSL       = when this word appear at the end of the destination string
#             the Netdata will encrypt the connection with the parent.
#
# This communication is not HTTP (it cannot be proxied by web proxies).
destination = A.B.C.D:19999
```

Fonte: Elaborado pelo autor

Inicialmente atribuímos ao campo `[stream].enabled = yes`. Ora, como mostrado na Sessão 2.4.1, trata-se de um nó secundário cuja funcionalidade é enviar seus dados por meio de *streaming* para um nó principal. Logo a opção de *streaming* deve estar habilitada em todos nós do *cluster*, excetuando-se o nó principal (STREAMING..., 2020a).

Ainda nesse arquivo de configuração, ativa-se a comunicação por chave *API* utilizando o valor da chave referenciada no nó principal da sessão anterior, que seria o mesmo valor do código da chave *API* gerado com o *uuidgen* na Sessão 2.4.2. Assim, adiciona-se *api key = 34319e67-1852-46ba-bd8c-6cf6ce6f2c0e* na linha abaixo do campo comentado "The API KEY to use" no arquivo de configuração (STREAMING..., 2020a).

Na configuração de *streaming* as métricas serão roteadas por meio do endereço de *IP* até o nó principal. Como na Sessão 3.2.3.1 referente ao mapeamento da rede foi atribuído o endereço de *A.B.C.D* ao nó principal e é substituído então esse valor no campo "destination =" do arquivo de configuração. De modo que temos *destination = A.B.C.D:19999* (STREAMING..., 2020a). Nota-se que deve ser inserida a porta da aplicação de modo a explicitar para a rede qual aplicação receberá o pacote de dados.

O arquivo mostrando os comparativos das configurações nativas e alterações promovidas para as sessões de interesse dos arquivos *.conf* referente aos nó secundários está no Apêndice F. Os parâmetros não alterados ou comentados são omitidos no arquivo.

Finalmente são repetidas as instruções dessa sessão em todas as máquinas constituintes da rede. A instalação segue a mesma estruturação, sendo necessário naturalmente iniciar a sessão no novo dispositivo a partir de uma conexão pré estabelecida com o servidor principal por meio do `ssh usuario@usuario0x`, sendo *x* o nó em questão.

3.2.4 Configuração da *dashboard* no *netdata*

Na Sessão 2.4.3 foi descrito qualitativamente o conceito de *dashboard* no *software netdata*, esta sessão tem como objetivo configurar a aplicação para as necessidades do projeto. Utiliza-se diretamente a implementação da ferramenta de *streaming*, de modo a criar uma interface gráfica própria que contenha parâmetros referentes a todos os nós do sistema de *clustering*.

Com a instalação do *software netdata* é gerado um arquivo em *JavaScript* referente às configurações da *dashboard* principal acessada por: `http://A.B.C.D:1999` (NETDATA..., 2020b). Este arquivo é o `dashboard_info.js` e é gerado no diretório *web*, isto é, `/usr/share/netdata/web/gui`. No entanto as mudanças aplicadas diretamente a esse arquivo não são persistentes, uma vez que são alteradas quando o *software* é atualizado, de modo que é atribuído nesse arquivo como parâmetro o conteúdo de um outro arquivo utilizado para customização: `dashboard_info_custom_example.js` (WEB..., 2020b).

De modo a manter o arquivo original como *backup* e modelo para eventuais futuras alterações na *dashboard* o arquivo original é copiado por meio do comando *CD*, referente à manipulação de arquivos em *linux*: `cd /usr/share/netdata/web/gui` e `sudo cp dashboard_info_custom_example.js lps_dashboard_info_file.js`.

Dessa forma, promovem-se as alterações desejadas no modelo do arquivo de configuração customizado copiado por meio de `sudo vim lps_dashboard_info_file.js`, o código no Apêndice H mostra trechos do código com exemplo de modificação utilizando de classes *JavaScript*. Nota-se que é possível customizar diferentes parâmetros utilizando diferentes classes de objetos. Ora, *JavaScript* é uma linguagem orientada a objeto de modo que é possível se utilizar de uma biblioteca de objetos para maior grau de customização (MULTI..., 2020).

Com a alteração efetuada, atribui-se ao *netdata* o arquivo `.js` que deve ser carregado quando a *dashboard* for inicializada, explicitado no corpo do código *JavaScript* por meio da alteração na sessão *web*: `[web]custom dashboard_info.js = lps_dashboard_info_file.js`, sendo `lps_dashboard_info_file.js` o arquivo copiado e customizado. Dessa forma o *software* reconhecerá a partir de qual arquivo `.js` deve carregar a *dashboard* (WEB..., 2020b).

Além disso, a customização não está restrita apenas à *dashboard* principal, ainda no diretório *web*, localizado em `/usr/share/netdata/web/` existe um protótipo de painel pré configurado para múltiplos nós, chamado *Multi-Host Dashboard*, sob o nome de `dash-example.html`. De forma análoga à customização implementada na *dashboard* principal o arquivo é copiado de modo a evitar sobre-escrita: `sudo cp /usr/share/netdata/web/dash-example.html /usr/share/netdata/web/lps.html`, logo, o arquivo `dash-example.html` foi copiado sob o nome `lps.html` (MULTI..., 2020).

Editam-se o arquivo *html* por meio de `sudo vim lps.html` e através da funcionalidade

de busca do editor, digita-se `/TUTORIAL`, uma vez que no corpo do código sob a *label* `/TUTORIAL` está indicada a parte de código em que deve-se editar de modo a viabilizar a obtenção de métricas por *streaming*, desse modo é localizado o parâmetro *var dash* na parte do código, definido como padrão por: `var dash = new Dash('http://localhost:19999')`. Dessa forma, é necessário modifica-lo para implementação para a rede local especificada no projeto `var dash = new Dash('http://A.B.C.D:19999')`. É Mostrado no Apêndice H o trecho destacado do código (MULTI..., 2020). Além disso, é importante alterar a permissão do arquivo de modo que o browse seja capaz de renderizar os gráficos a partir do código HTML. Dessa forma, como explicitado na Sessão 2.2.4 atribui-se permissão de grupo ao arquivo por meio do comando `sudo chmod 700 lps.html`.

Os gráficos são especificados pela sintaxe *dash-** e são mostradas para cada nó. Essa sintaxe é uma classe *html* aplicada em um *div* para plotar automaticamente os gráficos na página para cada nó. Ora, *div* é um contêiner genérico de conteúdo em *html*, é utilizado para agrupar conteúdos de modo que eles possam ser abordados e tratados globalmente usando uma classe ou outros atributos (MULTI..., 2020).

Por exemplo, no Apêndice J é mostrado um trecho de código do arquivo *html* de modo a explicitar o funcionamento da sintaxe. Define-se o agrupamento *div* incluindo parâmetros relacionados às métricas que se deseja exibir, por exemplo na linha `data-dash-netdata="system.cpu"` atribui-se à variável de dados (*data-dash-netdata*) métricas do tipo monitoramento *cpu*, de modo que as sintaxe das métricas utilizada arquivo *html* são tratadas pelos parâmetros declarados pela própria ferramenta como abordado na Sessão 2.4.3. No bloco *div* pode-se adicionar objetos do tipo *data-** indefinidamente, como por exemplo relacionadas à dimensão, título, cor e etc. Nota-se que qualquer parâmetro associado diretamente no bloco irá subscrever a configuração padrão, podendo ser atribuída diretamente assim qualquer valor desejado (MULTI..., 2020).

À esse bloco então é aplicada uma determinada classe, isto é, adiciona-se uma classe na sintaxe de sua inicialização `<div class="dash-graph"` (WEB..., 2020b). O que indica que esses parâmetros serão tratados pela classe de gráficos em linha, pode-se utilizar diferentes *layout* de tratamento gráficos definidos pelo próprio *netdata* (WEB..., 2020b). É possível ainda mudar o tamanho de exibição dos gráficos globalmente, para tal no editor *vim* basta procurar pelo objeto *Dash* com o comando `/‘Dash.options’` e alterar os seus valores para todos os gráficos (MULTI..., 2020).

Os objetos do tipo de métrica utilizada bem como as classes de gráficos disponíveis são definidas no próprio código fonte ou no arquivo *dashboard.js* neste mesmo diretório `/usr/share/netdata/web/gui` (WEB..., 2020b).

Todos os gráficos para todos os nós são plotados sob essa sintaxe. Ora, quando o arquivo *html* é carregado, sobre o objeto *dash* é aplicada uma outra classe mais abrangente que se utiliza do *netdata API* gerado na Sessão 2.4.2 de modo a localizar a lista de nós que

estão por meio de *streaming* enviando suas métricas para o nó principal, então as métricas são replicadas por meio dessa classe `<div class="netdata-host-stats-container template">` para cada um dos nós. Dessa forma, é obtido o grau de customização desejado por meio da substituição e adição de elementos com a sintaxe *dash-** de modo que as mudanças serão replicadas automaticamente para cada nó (MULTI..., 2020).

É importante salientar que a sintaxe *dash-** só funciona caso a ferramenta de *streaming* da Sessão 2.4.1 esteja efetivamente implementada (STREAMING..., 2020a). Além disso, o arquivo *html* (no caso *lps.html* só precisa ser configurada no nó principal, uma vez que esse nó receberá como parâmetro o *API* de toda rede local (MULTI..., 2020).

Com as modificações realizada é possível agora acessar diretamente o painel *multi-host dashboard* com todas as métricas desejadas e para todas as máquinas simultaneamente acessando diretamente o arquivo editado *lps.html* no *browser* com: `http://A.B.C.D:1999/lps.html` (MULTI..., 2020). Nota-se ainda que é possível acessar essa implementação diretamente pelo tunelamento em *SSH* como descrito detalhadamente na Sessão 2.2.7. Utiliza-se o próprio endereço da rede local como parâmetro para acesso no *browser* mesmo se conectado através de uma rede externa com outro *IP*, uma vez que a ferramenta de *streaming* atua localmente e o tunelamento encaminha todo tráfego local.

Adicionalmente, é fornecido um *script* em *bash* desenvolvido pelo autor de modo a automatizar o processo de configuração nos nós secundários, uma vez que tratam-se de 11 nós. O *script* disponível no Apêndice G é mostrado Na Figura 22 por meio de um *printscreen*. O *script* atua manipulando o redirecionamento de entrada e saída no terminal.

Figura 22 – *Script* de alteração nós secundários

```

1 sudo cat >/etc/netdata/netdata.conf <<EOF
2 [global]
3   memory mode = dbengine
4 [web]
5   mode = static-threaded
6 EOF
7
8 sudo cat >/etc/netdata/stream.conf <<EOF
9 [stream]
10  enabled = yes
11  destination = http://A.B.C.D:1999
12  api key = 34319e67-1852-46ba-bd8c-6cf6ce6f2c0e
13 EOF

```

Fonte: *Print* do *script* de instalação disponível no Apêndice G

Inicialmente na Figura 22 `>` precede o *pathname* absoluto do arquivo *netdata.conf* bem como posteriormente o arquivo *stream.conf* (MACH, 2020). O operador indica que é direcionado ao arquivo especificado qualquer comando que teria como saída padrão o terminal, isto é, a saída do comando será escrita no arquivo em vez de exibida no terminal. De forma análoga, a entrada também pode ser redirecionado por `<` (SHOTTS, 2019).

O operador na Figura 22 « indica que o arquivo deve ser lido até encontrar os parâmetro especificado indicado pelo bloco delimitada por *EndOfCommands* (*EOF*). Dessa forma, por meio do comando `cat` junto às ferramentas de manipulação de entrada e saída é possível alterar os dois arquivos de configuração com um *script* (SHOTTS, 2019). Nota-se que deve ser substituído no próprio *script* as linhas `destination = http://A.B.C.D:1999` e `34319e67-1852-46ba-bd8c-6cf6ce6f2c0e` por valores reais da especificidade de projeto.

4 Resultados

Nesta seção serão expostos os resultados obtidos na implementação da *dashboard* com a instalação e configuração do *software netdata*. O capítulo será dividido em três seções, uma referente ao acesso à *dashboard* principal, outra aos resultados da implementação da ferramenta de *streaming* e finalmente referente à implementação final do projeto, a *multi-dashboard* para exibição de todas as métricas. As subseções dividirão os algoritmos de otimização utilizados.

4.1 Resultados *Dashboard* principal

Com a instalação realizada pela Sessão 3.2.2, referente a instalação do *software* já é possível acessar a interface de análise de métricas do próprio programa. Para isso basta digitar no *browser* `http://localhost:19999`, sendo 19999 a porta de aplicação utilizada pelo *software netdata* e a variável *localhost* referente ao endereço *IP* estático de classe C do nó principal do *cluster*. Extensivamente empregado como `localhost = A.B.C.D` no decorrer do projeto.

Da mesma forma, para qualquer nó em que seja instalado o *software netdata* é possível acessar individualmente à *dashboard* principal por meio de `localhost = A.B.C.E:19999`, sendo "E" o último *byte* de endereçamento referente ao nó em questão, como explicitado na Sessão 2.1.2.1 por ser pertencente à mesma rede local do nó principal apenas esse campo é tratado como variável para o endereçamento, de modo que os nós diferem-se quanto ao valor associado à esse campo.

Nota-se que por meio das diretivas de configuração abordadas nesse projeto o acesso à *dashboard* principal não tem seu acesso limitado por meio da rede doméstica. Ora, conhecendo-se o *IP* real atribuído ao servidor *ssh* é possível acessá-la por meio de `http://X.Y.Z.W:19999`. Sendo X.Y.Z.W o endereço associado ao *ISP* do próprio servidor, isto é, antes do roteador realizar o *DHCP* para atribuição de endereçamento local, de acordo com o conteúdo apresentado na Sessão 2.1.2.1, sobre endereçamento *IP*. Nota-se ainda que esse endereço pode ser explicitado pelo sistema operacional por meio das diretrizes da Sessão 3.2.3.1, sobre mapeamento de rede.

Figura 23 – Acesso à *Dashboard* principal

Fonte: Elaborado pelo autor

4.1.1 Resultados Streaming

O acesso à *dashboard* principal é automaticamente configurada de modo a se obter métricas individualmente para a máquina em que se foi instalada a aplicação. No entanto, por meio da ferramenta de *streaming* foi possível conectar às aplicações *netdata* por meio da rede local (STREAMING..., 2020a). Ao realizar toda etapa de configuração da ferramenta na Sessão 3.2.3.2 será possível visualizar se a comunicação por *API* estiver efetivamente funcionando, isto é, por meio de `http://A.B.C.D:19999/api/v1/info`, de modo que os nós devem estar visíveis sob a notação, *JavaScript Object Notation (JSON)* e com sintaxe mirrored_host como mostrado na Figura 24. Os parâmetros *UID*, *GUID* bem como a versão do *Kernel* foram parcialmente omitidos de modo a possibilitar a visualização da saída sem identificação da rede.

Figura 24 – *Mirrored_host*

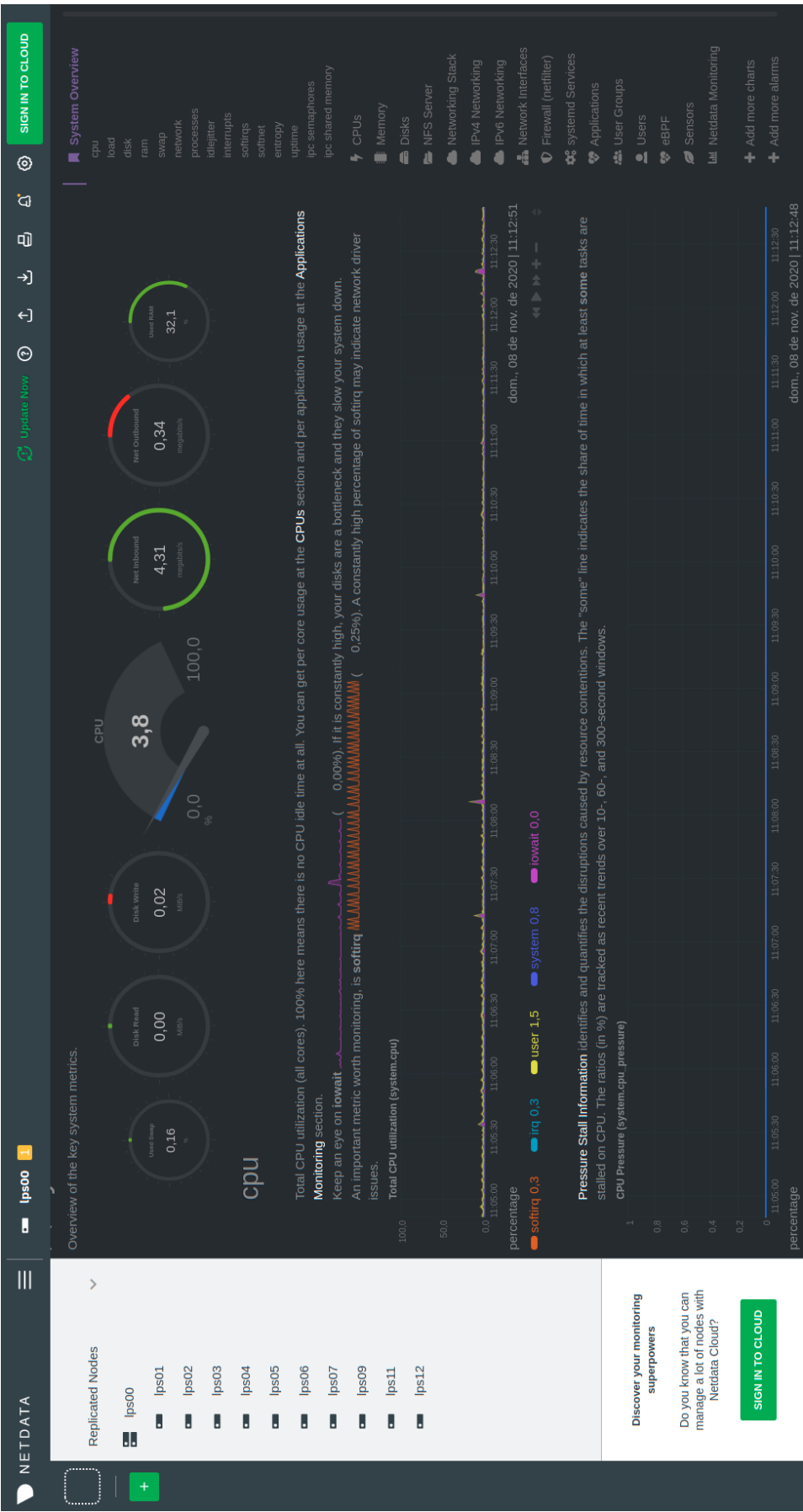
```
{
  "version": "v1.26.0-130-nightly",
  "uid": "dc8aeb-al 00e0 50b2",
  "mirrored_hosts": [
    "lps00",
    "lps08",
    "lps12",
    "lps11",
    "lps09",
    "lps07",
    "lps06",
    "lps05",
    "lps03",
    "lps02",
    "lps01",
    "lps04"
  ],
  "mirrored_hosts_status": [
    { "guid": "d", "4-2068-", "-00e", "i0b2", "reachable": true, "claim_id": "dc8af2",
    { "guid": "e", "4-fa5b-", "-04d", "lefd", "reachable": true, "claim_id": null },
    { "guid": "b", "4-2076-", "-34e", "93e", "reachable": true, "claim_id": null },
    { "guid": "d", "3-2075-", "-34e", "92f", "reachable": true, "claim_id": null },
    { "guid": "d", "3-2074-", "-34e", "80e", "reachable": true, "claim_id": null },
    { "guid": "9", "2-2071-", "-eda", "l102", "reachable": true, "claim_id": null },
    { "guid": "5", "3-2070-", "-c51", "i35f", "reachable": true, "claim_id": null },
    { "guid": "7", "3-206f-", "-391", "bbe", "reachable": true, "claim_id": null },
    { "guid": "3", "3-206d-", "-04d", "i3fc", "reachable": true, "claim_id": "33f8dc",
    { "guid": "8", "3-2064-", "-04d", "i5a1", "reachable": true, "claim_id": "81da10",
    { "guid": "0", "2-206a-", "-401", "i836", "reachable": true, "claim_id": null },
    { "guid": "1", "2-206f-", "-832", "i726", "reachable": true, "claim_id": "175e00"
  ],
  "alarms": {
    "normal": 99,
    "warning": 1,
    "critical": 0
  },
  "os_name": "Fedora",
  "os_id": "fedora",
  "os_id_like": "unknown",
  "os_version": "31 (Server Edition)",
  "os_version_id": "31",
  "os_detection": "/etc/os-release",
  "cores_total": "4",
  "total_disk_space": "4000797868032",
  "cpu_freq": "3700000000",
  "ram_total": "8319168512",
  "container_os_name": "none",
  "container_os_id": "none",
  "container_os_id_like": "none",
  "container_os_version": "none",
  "container_os_version_id": "none",
  "kernel_name": "Linux",
  "kernel_version": "5.4. 6_64",
  "architecture": "x86_64",
  "virtualization": "none",
  "virt_detection": "systemd-detect-virt",
  "container": "unknown",
  "container_detection": "none",
}
```

Fonte: Elaborado pelo autor

Dessa forma, é possível selecionar a *dashboard* de todos os nós por meio da *dashboard* do nó principal como mostrado na Figura 25. Ora, o nó principal é responsável por manter a base de dados remotamente devido às configurações de memória e servidor *web*

implementadas na Sessão 3.2.3.2, tomando como diretriz a Tabela 3 para as especificidades do projeto.

Figura 25 – Acesso a todas *Dashboard*



Fonte: Elaborado pelo autor

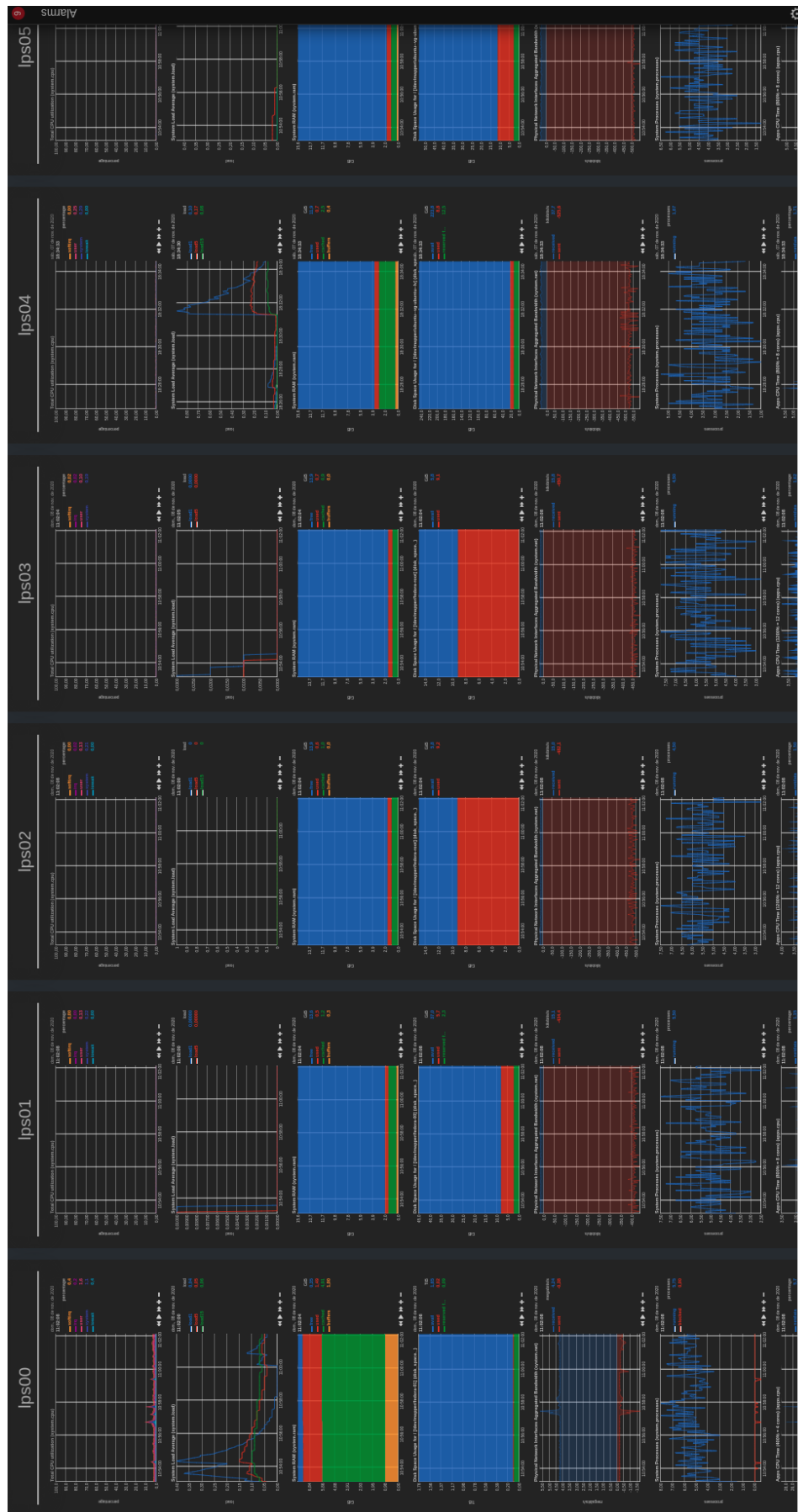
Logo, como mostrado na Figura 25 as métricas de todo o sistema *cluster* estão disponíveis no menu de seleção no canto esquerdo superior da *dashboard* principal. Assim todos os nós podem facilmente acessados alternadamente por meio da *dashboard* do nó principal. Nota-se ainda que devido às configurações da Sessão 3.2.3.2 o acesso pode ser feito em qualquer *browser* tanto a partir da rede local com `http://A.B.C.D:19999` quanto remotamente a partir de qualquer dispositivo com acesso a um *browser* e conexão com a internet por meio de `http://X.Y.Z.W:19999` sem necessidade de *SSH tunnel*.

4.1.2 Resultados *Multi-Dashboard*

Diretamente por meio da implementação da ferramenta de *streaming* é possível gerar uma interface gráfica própria e customizadas contendo os parâmetros selecionados referentes às métricas de todos os nós do sistema de *clustering*. Na Sessão 3.2.4 foi alterado o código em *JavaScript* de modo a se exibir as métricas simultaneamente por meio de uma única interface, com a criação do arquivo *lps.html*.

De forma análoga ao acesso da *dashboard* principal basta digitar no *browser* `http://localhost:19999/lps.html`, com *localhost* = *A.B.C.D* sendo o *IP* estático de classe C do nó principal do *cluster*. Ora, a página carregada consiste de uma implementação *JavaScript* sendo executada na máquina principal no diretório `/usr/share/netdata/web/lps.html`, logo quando o *browser* realizar uma requisição de *URL* para carregar a página as métricas serão obtidas localmente pela *info API* a partir da *JSON* sob sintaxe `mirroed_host`. Dessa forma, os gráficos para os parâmetros de interesse são escolhidos por meio da modificação no arquivo *lps.html* como mostrado na Sessão 3.2.4 e são plotados recursivamente por meio da sintaxe *dash-**. O resultado final da implementação é mostrado na Figura 26.

Figura 26 – Multi-Dashboard para todas máquinas



Fonte: Elaborado pelo autor

5 Conclusão

A implementação da ferramenta de *streaming*, bem como o mapeamento da rede foram bem sucedidas em obter parâmetros necessários para viabilizar os acesso às *dashboards*. A implementação e configuração do *software Netdata* no *cluster Beowulf* do projeto forneceu informativos sobre o desempenho em tempo real de praticamente todos parâmetros de interesse de forma customizada, individualmente são exibidas as *dashboard* de cada *host* com menu para navegação rápida entre toda a rede, permitindo maior granularidade na detecção de anomalias devido às diretivas de configurações abordadas. Nota-se que não é necessário tunelamento de *proxy* para essa acessibilidades devido á estruturação do servidor *SSH* do nó principal do *cluster* e configuração do *Netdata*.

O resultado final pode ser acessado de forma análoga ao acesso da *dashboard* principal por meio da rede interna, basta especificar a diretriz *html*. No entanto, essa implementação não é possível de ser acessada diretamente com *IP "real"* do *cluster* como a configuração das *dashboard* individuais realizada na primeira etapa do projeto. Uma vez que ao se acessar o *browser* o código JS do *lps.html* tentará plotador recursivamente por meio da sintaxe *dash-** tendo como argumento a *LAPI* de *localhost*.

Nota-se no entanto, que uma vez conectado á rede interna por meio de um *tunnel proxy SSH*, todos os resultados podem ser obtidos normalmente, como se estivesse presente fisicamente no laboratório LPS, também é possível utilizar outros serviços como *OpenVPN* e acessar remotamente a rede local de forma segura.

Sugere-se como futuras implementações a configuração de outros *softwares* coletores de métricas que são integráveis ao *netdata*, uma lista de *softwares* compatíveis é disponibilizada na própria documentação do *netdata*¹. Também é possível explorar a estruturação *cloud*² recém implementada pela equipe *netdata*. Ou então ainda implementar o gerenciamento e monitoramento de métricas para o âmbito de *IoT*³, com automatização no gerenciamento, isto é especificar configuração de coletores de modo a enviar á alguma plataforma ou serviço compatível com o *netdata* notificações sobre uso computacional do hardware por *e-mail* ou outras notificações.

¹Disponível em <<https://learn.netdata.cloud/docs/agent/collectors/collectors>>

²Disponível em <<https://learn.netdata.cloud/docs/cloud>>

³Disponível em <<https://learn.netdata.cloud/docs/agent/netdata-for-iot>>

Referências

- 5G, IEEE 802.11ax. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <<https://rf.eefocus.com/article/id-332918>>. Citado na página 35.
- ABBASI, A. A.; YOUNIS, M. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, v. 30, n. 14, p. 2826 – 2841, 2007. ISSN 0140-3664. Network Coverage and Routing Schemes for Wireless Sensor Networks. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0140366407002162>>. Citado 2 vezes nas páginas 28 e 31.
- ABBOTT, D. *Linux for Embedded and Real-time Applications*. [S.l.]: Newnes, 2013. (Embedded technology series). ISBN 978-0-12-415996-9, 9780123914330, 0123914337. Citado 4 vezes nas páginas 46, 77, 79 e 90.
- ADAMS, J.; VOS, D. Small-college supercomputing: Building a beowulf cluster at a comprehensive college. In: *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2002. (SIGCSE '02), p. 411–415. ISBN 1581134738. Disponível em: <<https://doi.org/10.1145/563340.563498>>. Citado 2 vezes nas páginas 23 e 25.
- ADAMS, J.; VOS, D. Small-college supercomputing: Building a beowulf cluster at a comprehensive college. *SIGCSE Bull.*, Association for Computing Machinery, New York, NY, USA, v. 34, n. 1, p. 411–415, fev. 2002. ISSN 0097-8418. Disponível em: <<https://doi.org/10.1145/563517.563498>>. Citado na página 63.
- AGENT Database. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/database>>. Citado 2 vezes nas páginas 68 e 69.
- AHLWEDE, R. et al. Network information flow. *IEEE Transactions on Information Theory*, v. 46, n. 4, p. 1204–1216, 2000. Cited By 6471. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-0034229404&doi=10.1109%2F18.850663&partnerID=40&md5=92607d583169b35ae6511e27fc34164c>>. Citado na página 31.
- AKHILA, K.; GANESH, A.; SUNITHA, C. A study on deduplication techniques over encrypted data. *Procedia Computer Science*, v. 87, p. 38 – 43, 2016. ISSN 1877-0509. Fourth International Conference on Recent Trends in Computer Science Engineering (ICRTCSE 2016). Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050916304628>>. Citado 2 vezes nas páginas 67 e 69.
- ALPERN, N. J.; SHIMONSKI, R. J. Tcp/ip and routing. In: ALPERN, N. J.; SHIMONSKI, R. J. (Ed.). *Eleventh Hour Network+*. Boston: Syngress, 2010. p. 89 – 105. ISBN 978-1-59749-428-1. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9781597494281000011>>. Citado 4 vezes nas páginas 37, 38, 43 e 86.
- Amiri, I. S. et al. Generation of discrete frequency and wavelength for secured computer networks system using integrated ring resonators. 2012. Citado 2 vezes nas páginas 27 e 34.

BALASUBRAMANIAN, A. Parameterized verification of coverability in infinite state broadcast networks. *Information and Computation*, Elsevier, 2020. Citado 3 vezes nas páginas 37, 41 e 43.

BARRETT DANIEL J.; SILVERMAN, R. E. *SSH, The Secure Shell: The Definitive Guide*. [S.l.]: O'Reilly Media, Inc, 2009. ISBN 0596000111. Citado 8 vezes nas páginas 45, 46, 78, 79, 81, 84, 89 e 94.

BLUM, C. B. R. *Linux Command Line and Shell Scripting Bible*. 4rd. ed. [S.l.]: Wiley, 2020. ISBN 978-1118983843. Citado 5 vezes nas páginas 47, 48, 49, 50 e 61.

BOTH, D. *Using and Administering Linux: Volume 3: Zero to SysAdmin: Network Services*. 1. ed. [S.l.]: Apress, 2020. v. 3. ISBN 1484254848,9781484254844. Citado 4 vezes nas páginas 46, 47, 62 e 85.

BRESNAHAN, R. B. C. *LPI Linux Essentials Study Guide: Exam 010 v1.6*. 3. ed. [S.l.]: Sybex, 2020. ISBN 1119657695,9781119657699. Citado na página 80.

C., D. T. Scalable file replication and web-based access. In: . [s.n.], 2015. Disponível em: <<https://patents.google.com/patent/US7743023B2/en>>. Citado 2 vezes nas páginas 72 e 92.

CALLOC. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <<https://documentation.help/C-Cpp-Reference/calloc.html>>. Citado na página 70.

CHAE, B. K. The evolution of the internet of things (iot): A computational text analysis. *Telecommunications Policy*, v. 43, n. 10, p. 101848, 2019. ISSN 0308-5961. ITS Seoul 2018. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0308596118303094>>. Citado na página 43.

CHALLOO, R. et al. An overview and assessment of wireless technologies and co-existence of zigbee, bluetooth and wi-fi devices. *Procedia Computer Science*, v. 12, p. 386 – 391, 2012. ISSN 1877-0509. Complex Adaptive Systems 2012. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050912006825>>. Citado 3 vezes nas páginas 28, 36 e 37.

CHAPPLE, M. *SSH2*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://searchsecurity.techtarget.com/tip/An-introduction-to-SSH2>>. Citado 7 vezes nas páginas 51, 52, 53, 54, 55, 56 e 59.

COOK, S. Chapter 11 - designing gpu-based systems. In: COOK, S. (Ed.). *CUDA Programming*. Boston: Morgan Kaufmann, 2013, (Applications of GPU Computing Series). p. 503 – 526. ISBN 978-0-12-415933-4. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780124159334000119>>. Citado na página 64.

COREUTILS Software®. 2020. Acesso em: 15 de nov. de 2020. Disponível em: <<https://www.gnu.org/software/coreutils>>. Citado na página 47.

DAHILI, A. Beowulf-class computer system at msu-iit. In: . [S.l.: s.n.], 2001. Citado na página 63.

DANIEL, L. E.; DANIEL, L. E. Chapter 22 - discovery of internet service provider records. In: DANIEL, L. E.; DANIEL, L. E. (Ed.). *Digital Forensics for Legal Professionals*. Boston: Syngress, 2012. p. 151 – 156. ISBN 978-1-59749-643-8. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9781597496438000225>>. Citado na página 39.

DATTI, A. A.; UMAR, H. A.; GALADANCI, J. A beowulf cluster for teaching and learning. In: . [s.n.], 2015. v. 70, p. 62 – 68. ISSN 1877-0509. Proceedings of the 4th International Conference on Eco-friendly Computing and Communication Systems. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050915031981>>. Citado 2 vezes nas páginas 23 e 63.

DBENGINE. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://www.netdata.cloud/blog/db-engine/>>. Citado 6 vezes nas páginas 66, 67, 68, 69, 70 e 91.

D.E., C. *Internetworking with TCP/IP: Principles, Protocols, and Architecture (Internetworking with TCP/IP)*. 4. ed. [S.l.: s.n.], 2000. Volume 1. Citado 4 vezes nas páginas 29, 36, 37 e 86.

DEAL, R. *Cisco router firewall security*. [S.l.]: Cisco Press, 2004. ISBN 1587051753,9781587051753. Citado 3 vezes nas páginas 44, 59 e 88.

DIKE, J. *inst*. [S.l.]: Prentice Hall, 2006. ISBN 0131865056,9780131865051. Citado na página 75.

DOCS Fedora Project. Disponível em: <<https://docs.fedoraproject.org/en-US/docs/>>. Citado 2 vezes nas páginas 80 e 87.

DUATO, J.; YALAMANCHILI, S.; NI, L. Message switching layer. In: DUATO, J.; YALAMANCHILI, S.; NI, L. (Ed.). *Interconnection Networks*. San Francisco: Morgan Kaufmann, 2003, (The Morgan Kaufmann Series in Computer Architecture and Design). p. 43 – 81. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9781558608528500055>>. Citado na página 33.

EDIT Configure. 2020. Acesso em: 03 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/configure/nodes#use-edit-config-to-edit-netdataconf>>. Citado na página 90.

ELTRINOS, F. *Kali Linux: Testing Your Network: How to Test Infrastructure Security with Security Testing and Penetration Testing*. [S.l.: s.n.], 2020. Citado 4 vezes nas páginas 46, 47, 85 e 87.

EPL. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.redhat.com/en/blog/whats-epel-and-how-do-i-use-it>>. Citado na página 62.

FABRICIUS, U. et al. High performance computing education for students in computational engineering. In: SUNDERAM, V. S. et al. (Ed.). *Computational Science – ICCS 2005*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 27–35. ISBN 978-3-540-32114-9. Citado na página 23.

FARREL, A. Chapter 2 - the internet protocol. In: FARREL, A. (Ed.). *The Internet and Its Protocols*. Burlington: Morgan Kaufmann, 2004, (The Morgan Kaufmann Series in Networking). p. 23 – 77. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781558609136500224>>. Citado 2 vezes nas páginas 38 e 39.

FERREIRA, L. et al. Cloudlet architecture for dashboard in cloud and ubiquitous manufacturing. *Procedia CIRP*, v. 12, p. 366 – 371, 2013. ISSN 2212-8271. Eighth CIRP Conference on Intelligent Computation in Manufacturing Engineering. Disponível em: <http://www.sciencedirect.com/science/article/pii/S221282711300704X>>. Citado na página 65.

Fu, G. The research on transmission of networked control based on ip switching. In: *2011 International Conference on Electric Information and Control Engineering*. [S.l.: s.n.], 2011. p. 3736–3739. Citado na página 32.

GAO, S. et al. Api recommendation for the development of android app features based on the knowledge mined from app stores. *Science of Computer Programming*, v. 202, p. 102556, 2021. ISSN 0167-6423. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167642320301647>>. Citado 3 vezes nas páginas 68, 72 e 89.

GIFT, J. J. N. *Python for Unix and Linux System Administration [Noah Gift] (2009)*. [S.l.: s.n.], 2008. ISBN 0596515820,978-0-596-51582-9. Citado na página 68.

GIT distributed version control. 2020. Acesso em: 10 de nov. de 2020. Disponível em: <https://git-scm.com>>. Citado na página 80.

GITHUB Netdata. Disponível em: <https://github.com/netdata/netdata>>. Citado 2 vezes nas páginas 77 e 80.

GONG, D.; ZHAO, M.; YANG, Y. Distributed channel assignment algorithms for 802.11n wlans with heterogeneous clients. *Journal of Parallel and Distributed Computing*, v. 74, n. 5, p. 2365 – 2379, 2014. ISSN 0743-7315. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0743731514000197>>. Citado 2 vezes nas páginas 28 e 36.

GROPP EWING LUSK, T. S. J. H. W. *Beowulf cluster computing with Linux*. 2. ed. [S.l.]: The MIT Press, 2003. (Scientific and Engineering Computation). ISBN 9780262692922,0262692929. Citado 7 vezes nas páginas 24, 25, 44, 79, 84, 85 e 86.

GROPP, W.; THAKUR, R.; BALAJI, P. Translational research in the mpich project. *Journal of Computational Science*, p. 101203, 2020. ISSN 1877-7503. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877750320305044>>. Citado na página 24.

GROSS, J. et al. Enhancing ieee 802.11a/n with dynamic single-user ofdm adaptation. *Performance Evaluation*, v. 66, n. 3, p. 240 – 257, 2009. ISSN 0166-5316. Modeling and Analysis of Wireless Networks: Selected Papers from MSWiM 2007. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166531608000941>>. Citado na página 36.

HAGEN. *The Definitive Guide to GCC*. 2. ed. [S.l.]: Apress, 2006. ISBN 978-1-59059-585-5,1-59059-585-8. Citado na página 80.

HARRINGTON, J. L. *2 - How TCP/IP and Ethernet Work*. Burlington: Morgan Kaufmann, 2007. 21 - 38 p. ISBN 978-0-12-373744-1. Citado na página 24.

HAYKIN, S. *Communication Systems*. 4th ed. ed. [S.l.]: Wiley, 2001. ISBN 0-471-17869-1. Citado 4 vezes nas páginas 31, 32, 34 e 36.

HEALTH Alarm. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/health/reference/#alarm-line-on>>. Citado na página 66.

Hu, S.; Zhu, Y.; Xiao, X. Performance research of iee 802.11 wlan. In: *2013 International Conference on Computational and Information Sciences*. [S.l.: s.n.], 2013. p. 1323–1326. Citado na página 36.

HYPERVISOR. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor>>. Citado na página 69.

INFRASTRUCTURE OpenSSH. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://docs.fedoraproject.org/en-US/fedora/rawhide/system-administrators-guide/infrastructure-services/OpenSSH/>>. Citado 10 vezes nas páginas 46, 50, 52, 53, 54, 55, 56, 57, 58 e 59.

INSAM, E. Network layer – building on ip. In: INSAM, E. (Ed.). *TCP/IP Embedded Internet Applications*. Oxford: Newnes, 2003. p. 171 – 206. ISBN 978-0-7506-5735-8. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780750657358500345>>. Citado 3 vezes nas páginas 37, 40 e 43.

INSTALLER Netdata. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/packaging/installer>>. Citado 10 vezes nas páginas 27, 79, 80, 81, 82, 83, 87, 88, 89 e 90.

IOT Netdata docs. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://github.com/netdata/netdata/blob/master/docs/netdata-for-IoT.md>>. Citado na página 66.

J., G. et al. Data consistency matrix based data processing model for efficient data storage in wireless sensor networks. *Computer Communications*, v. 151, p. 172 – 182, 2020. ISSN 0140-3664. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S014036641930502X>>. Citado na página 67.

Kai Shen; Tao Yang; Lingkun Chu. Cluster load balancing for fine-grain network services. p. 8 pp–, 2002. Citado 2 vezes nas páginas 23 e 24.

KENNINGTON, C. *Clusmon: A Beowulf Cluster Monitor*. [S.l.]: Boise State University, 2006. Citado na página 26.

KENNINGTON, C. Clusmon: A beowulf cluster monitor. Computer science department Boise State university, 2006. Citado na página 26.

Khattak, M. K. et al. Effective routing technique: Augmenting data center switch fabric performance. *IEEE Access*, v. 8, p. 37372–37382, 2020. Citado na página 32.

KICKSTART Command. Disponível em: <<https://pykickstart.readthedocs.io/en/latest/kickstart-docs.html/#chapter-2-kickstart-commands-in-fedora>>. Citado na página 83.

KICKSTART Netdata installer. 2020. Acesso em: 10 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/packaging/installer/methods/kickstart>>. Citado 2 vezes nas páginas 83 e 84.

KIDWAI, A. et al. A comparative study on shells in linux: A review. *Materials Today: Proceedings*, 2020. ISSN 2214-7853. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2214785320363902>>. Citado 2 vezes nas páginas 46 e 78.

KIM, D. A 2020 perspective on “a dynamic model for the evolution of the next generation internet – implications for network policies”: Towards a balanced perspective on the internet’s role in the 5g and industry 4.0 era. *Electronic Commerce Research and Applications*, v. 41, p. 100966, 2020. ISSN 1567-4223. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1567422320300430>>. Citado na página 32.

KUROSE JAMES F.; ROSS, K. W. *Computer Networking : A top-down approach*. 7. ed. [S.l.: s.n.], 2017. ISBN 978-85-52971-00-9. Citado 10 vezes nas páginas 23, 31, 32, 37, 40, 41, 43, 59, 60 e 61.

KVM. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://www.redhat.com/en/topics/virtualization/what-is-KVM>>. Citado na página 69.

LEARN Netdata. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud>>. Citado 6 vezes nas páginas 26, 62, 65, 66, 75 e 77.

LIB Paramater. 2020. Acesso em: 13 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/libnetdata/config>>. Citado na página 89.

LINUX Foundation. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.linuxfoundation.org/>>. Citado na página 62.

LIPOR, J.; BALZANO, L. Clustering quality metrics for subspace clustering. *Pattern Recognition*, v. 104, p. 107328, 2020. ISSN 0031-3203. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S003132032030131X>>. Citado na página 26.

LSB. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://wiki.linuxfoundation.org/>>. Citado na página 62.

M., E.-R. H. A.-E.-B. *Fundamentals of Computer Organization and Architecture*. [S.l.]: Wiley, 2005. (Wiley series on parallel and distributed computing). ISBN 0471467405,0471467413. Citado 4 vezes nas páginas 50, 67, 68 e 69.

MACH, D. *COMPUTER PROGRAMMING: LINUX COMMAND-LINE PYTHON Programming*. 1. ed. [S.l.]: Wiley, 2020. v. 1. ISBN 9798651465019. Citado 14 vezes nas páginas 46, 47, 48, 49, 50, 52, 53, 54, 56, 60, 61, 62, 88 e 98.

MALIK, A. et al. Qos in ieee 802.11-based wireless networks: A contemporary review. *Journal of Network and Computer Applications*, v. 55, p. 24 – 46, 2015. ISSN 1084-8045. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1084804515000892>>. Citado 2 vezes nas páginas 36 e 37.

MASTER Daemon Config. 2020. Acesso em: 10 de nov. de 2020. Disponível em: <https://github.com/netdata/netdata/tree/master/daemon/config>. Citado 3 vezes nas páginas 81, 89 e 90.

Frequency allocations for industrial, scientific, and medical (ism) applications. In: MEHDIZADEH, M. (Ed.). *Microwave/RF Applicators and Probes (Second Edition)*. Second edition. Boston: William Andrew Publishing, 2015. p. 369 – 370. ISBN 978-0-323-32256-0. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780323322560000166>. Citado na página 36.

MORIARTY K., E. K. B.-J. J.; RUSCH, A. *PKCS: RSA Cryptography Specifications Version 2.2*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <http://www.rfc-editor.org/info/rfc8017>. Citado na página 53.

MOTTA, M. Internet backbone topology in brazil. *Sociedade Natureza*, v. 24, p. 21–35, 04 2012. Citado 3 vezes nas páginas 32, 37 e 39.

MULTI Dashboards overview. 2020. Acesso em: 12 de nov. de 2020. Disponível em: <https://learn.netdata.cloud/docs/agent/web/gui/custom>. Citado 5 vezes nas páginas 26, 27, 96, 97 e 98.

MULTIPLE Instances OpenSSH. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <https://access.redhat.com/solutions/1166283>. Citado 3 vezes nas páginas 55, 58 e 59.

NEGUS, C. *Linux Bible*. 10th edition. ed. [S.l.]: WileySons, 2020. ISBN 1119578884,9781119578888,1119578914,9781119578918. Citado 17 vezes nas páginas 23, 25, 27, 41, 44, 46, 49, 50, 53, 54, 56, 59, 61, 77, 81, 90 e 93.

NETDATA Agent. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <https://learn.netdata.cloud/docs/agent/daemon>. Citado na página 82.

NETDATA Basics. 2020. Acesso em: 03 de nov. de 2020. Disponível em: <https://learn.netdata.cloud/docs/agent/getting-started>. Citado na página 96.

NETDATA Packages. 2020. Acesso em: 10 de nov. de 2020. Disponível em: <https://learn.netdata.cloud/docs/agent/packaging/installer/methods/manual>. Citado 2 vezes nas páginas 62 e 63.

Nguyen, D. et al. Wireless broadcast using network coding. *IEEE Transactions on Vehicular Technology*, v. 58, n. 2, p. 914–925, 2009. Citado 2 vezes nas páginas 37 e 86.

NMAP Install. 2020. Acesso em: 03 de nov. de 2020. Disponível em: <https://nmap.org/book/install.html>. Citado 4 vezes nas páginas 60, 68, 87 e 88.

NUECHTERLEIN, P. J. W. J. E. *Digital crossroads: American telecommunications policy in the Internet age*. 1. ed. [S.l.]: The MIT Press, 2005. ISBN 0262140918,9780262140911,026264066X,9780262640664,0262280787,9780262280785. Citado 3 vezes nas páginas 24, 37 e 39.

OPEN Sourcer. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <https://www.netdata.cloud/blog/open-source-contributions/>. Citado na página 66.

OPEN Sourcer Contributions. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <https://learn.netdata.cloud/docs/agent/contributing>. Citado na página 66.

- PATIL, A. *Socks-Proxy SOCKS5*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://securityintelligence.com/posts/socks-proxy-primer-what-is-socks5-and-why-should-you-use-it/>>. Citado 2 vezes nas páginas 60 e 61.
- PERLMAN, R. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. 2. ed. [S.l.]: Addison-Wesley Professional, 1999. ISBN 0201634481,9780201634488. Citado 5 vezes nas páginas 31, 32, 40, 43 e 86.
- PETERSON, B. S. D. L. L. *Computer Networks: A Systems Approach*. 3. ed. [S.l.]: Morgan Kaufmann, 2003. (The Morgan Kaufmann Series in Networking). ISBN 155860832X,9781558608320,9780080488547. Citado 13 vezes nas páginas 23, 24, 25, 28, 31, 32, 33, 34, 37, 38, 39, 41 e 87.
- PKCS: RSA Cryptography Specifications Version 2.2. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.rfc-editor.org/info/rfc8017>>. Citado na página 53.
- PROJECT, F. *Package EPEL.3F*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <https://fedoraproject.org/wiki/EPEL#What_packages_and_versions_are_available_in_EPEL.3F>. Citado na página 63.
- RED hat acquisition. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <<https://www.redhat.com/en/about/press-releases/ibm-closes-landmark-acquisition-red-hat-34-billion-defines-open-hybrid-cloud-future>>. Citado na página 45.
- RED Hat® Enterprise Linux®. 2020. Acesso em: 15 de nov. de 2020. Disponível em: <<https://www.redhat.com>>. Citado na página 45.
- RELEASE V1.15.0 Netdata. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <<https://github.com/netdata/netdata/releases/tag/v1.15.0>>. Citado na página 67.
- RESOURCER Type. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <<https://ant.apache.org/manual/Types/resources.html>>. Citado na página 69.
- SANDERS, C. *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems*. 3. ed. [S.l.]: No Starch Press, 2017. ISBN 1593272669,9781593272661. Citado na página 77.
- SEDRAKYAN, G.; MANNENS, E.; VERBERT, K. Guiding the choice of learning dashboard visualizations: Linking dashboard design and data visualization concepts. *Journal of Computer Languages*, v. 50, p. 19 – 38, 2019. ISSN 2590-1184. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1045926X18301009>>. Citado na página 65.
- SERMERSHEIM, E. J. *LDAP*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://tools.ietf.org/rfc/rfc4511.txt>>. Citado na página 57.
- SHOTTS, J. W. E. *The Linux Command Line*. 2. ed. [S.l.]: No starsh press, 2019. ISBN 9781593279523. Citado 10 vezes nas páginas 49, 50, 53, 54, 61, 62, 72, 87, 98 e 99.
- SIMPLE Pattern. 2020. Acesso em: 03 de nov. de 2020. Disponível em: <https://learn.netdata.cloud/docs/agent/libnetdata/simple_pattern>. Citado na página 92.

- SNIA Storage Management. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://www.snia.org/technology-focus/storage-management>>. Citado na página 67.
- SOBELL, M. G. *Practical Guide to Fedora and Red Hat Enterprise Linux, A (7th Edition)*. 7. ed. [S.l.]: Pearson Education, 2013. ISBN 978-0133477436, 0133477436. Citado na página 87.
- SPIVAK, A. et al. Storage tier-aware replicative data reorganization with prioritization for efficient workload processing. *Future Generation Computer Systems*, v. 79, p. 618 – 629, 2018. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X17305502>>. Citado na página 67.
- SRINIVAS, C.; RADHAKRISHNA, V.; RAO, C. G. Clustering and classification of software component for efficient component retrieval and building component reuse libraries. *Procedia Computer Science*, v. 31, p. 1044 – 1050, 2014. ISSN 1877-0509. 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050914005353>>. Citado na página 25.
- STERLING DANIEL F. SAVARESE, D. J. B. J. S. T. *How to Build a Beowulf*. [S.l.]: The MIT Press, 1999. (Scientific and Engineering Computation). ISBN 026269218X, 9780262692182, 9780585087450. Citado 6 vezes nas páginas 23, 24, 25, 26, 28 e 64.
- STERLING, T.; ANDERSON, M.; BRODOWICZ, M. Commodity clusters. In: STERLING, T.; ANDERSON, M.; BRODOWICZ, M. (Ed.). *High Performance Computing*. Boston: Morgan Kaufmann, 2018. p. 83 – 114. ISBN 978-0-12-420158-3. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780124201583000034>>. Citado na página 64.
- STERLING, T. et al. Beowulf: A parallel workstation for scientific computation. In: *ICPP*. [S.l.: s.n.], 1995. Citado na página 63.
- STEVENS BILL FENNER, A. M. R. W. R. *Unix Network Programming, Volume 1: The Sockets Networking API*. 3. ed. [S.l.]: Addison-Wesley Professional, 2003. v. 1. ISBN 0131411551, 9780134900124, 9780131411555, 013490012X. Citado na página 72.
- STREAMING and replication | Learn Netdata. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/streaming>>. Citado 17 vezes nas páginas 27, 28, 65, 70, 71, 72, 76, 85, 88, 89, 90, 91, 92, 94, 95, 98 e 103.
- STREAMING tree. 2020. Acesso em: 12 de nov. de 2020. Disponível em: <<https://github.com/netdata/netdata/tree/master/streaming>>. Citado 3 vezes nas páginas 27, 93 e 94.
- SWAP Memory. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <https://linuxhint.com/swap_memory_linux/>. Citado na página 69.
- SYSTEM, D. O. *Stack Overflow*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://insights.stackoverflow.com/survey/2016#technology-desktop-operating-system>>. Citado na página 44.

TANENBAUM, A. S. *Computer networks*. 4. ed. [S.l.]: Prentice Hall, 2002. Citado 7 vezes nas páginas 31, 32, 33, 34, 38, 39 e 43.

TECHNOLOGY-DESKTOP-OPERATING-SYSTEM. 2020. Acesso em: 03 de nov. de 2020. Disponível em: <<https://insights.stackoverflow.com/survey/2016#technology-desktop-operating-system>>. Citado na página 25.

THEPUATRAKUL, T. Ethernet-based interconnections for massively parallel clusters. In: . 1518 Piboonsongkram Rd., Bangsue Bangkok 10800, Thailand.: King Mongkut's Institute of Technology Bangkok, 2010. Citado na página 63.

TOURNIER, J. et al. A survey of iot protocols and their security issues through the lens of a generic iot stack. *Internet of Things*, p. 100264, 2020. ISSN 2542-6605. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2542660520300986>>. Citado na página 43.

Tschinkel, G. et al. The recommendation dashboard: A system to visualise and organise recommendations. In: *2015 19th International Conference on Information Visualisation*. [S.l.: s.n.], 2015. p. 241–244. Citado na página 64.

UPTADER Netdata. 2020. Acesso em: 10 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/packaging/installer/update>>. Citado na página 83.

UTHAYOPAS, P.; MANEESILP, J.; INGONGNAM, P. Scms: An integrated cluster management tool for beowulf cluster system. In: CITESEER. *PDPTA*. [S.l.], 2000. Citado na página 26.

VASSEUR, J.-P.; PICKAVET, M.; DEMEESTER, P. Ip routing. In: VASSEUR, J.-P.; PICKAVET, M.; DEMEESTER, P. (Ed.). *Network Recovery*. Burlington: Morgan Kaufmann, 2004, (The Morgan Kaufmann Series in Networking). p. 203 – 295. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780127150512500181>>. Citado 2 vezes nas páginas 40 e 41.

VERSTOEP, K. et al. Cluster communication protocols for parallel-programming systems. *ACM Trans. Comput. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 3, p. 281–325, ago. 2004. ISSN 0734-2071. Disponível em: <<https://doi.org/10.1145/1012268.1012269>>. Citado na página 23.

VIRTUALIZATION. 2020. Acesso em: 08 de nov. de 2020. Disponível em: <<https://www.redhat.com/en/topics/virtualization/what-is-virtualization-management>>. Citado na página 69.

Wang, L. et al. Linux kernels as complex networks: A novel method to study evolution. In: *2009 IEEE International Conference on Software Maintenance*. [S.l.: s.n.], 2009. p. 41–50. Citado 6 vezes nas páginas 24, 41, 46, 50, 57 e 61.

WEB Archive Fedora. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <<https://web.archive.org/web/20031001204515/http://www.fedora.us/>>. Citado na página 45.

WEB dashboards overview. 2020. Acesso em: 02 de nov. de 2020. Disponível em: <<https://learn.netdata.cloud/docs/agent/web>>. Citado 7 vezes nas páginas 26, 27, 73, 75, 77, 96 e 97.

WHITE, E. B. R. *Computer Networking Problems and Solutions: An innovative approach to building resilient, modern networks*. 1. ed. [S.l.]: Addison-Wesley Professional, 2018. ISBN 1587145049,9781587145049. Citado 13 vezes nas páginas 31, 32, 33, 37, 38, 39, 40, 41, 42, 43, 44, 59 e 85.

WI-FI enable devices. 2020. Acesso em: 09 de nov. de 2020. Disponível em: <<https://www.researchandmarkets.com/reports/4826074/global-wi-fi-enabled-devices-shipment-forecast>>. Citado na página 28.

YAN, Z.; LEE, J.-H. Mobility capability negotiation for ipv6 based ubiquitous mobile internet. *Computer Networks*, v. 157, p. 24 – 28, 2019. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128618311496>>. Citado na página 43.

YIGITBASIOGLU, O. M.; VELCU, O. A review of dashboards in performance management: Implications for design and research. *International Journal of Accounting Information Systems*, v. 13, n. 1, p. 41 – 59, 2012. ISSN 1467-0895. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1467089511000443>>. Citado 2 vezes nas páginas 64 e 65.

YLONEN, T. *Key Gen*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.ssh.com/ssh/keygen>>. Citado na página 57.

YLONEN, T. *Key SSH*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.ssh.com/ssh/public-key-authentication>>. Citado 7 vezes nas páginas 50, 51, 52, 54, 55, 57 e 59.

YLONEN, T. *Manual OpenSSH*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.openssh.com/manual.html>>. Citado na página 53.

YLONEN, T. *OpenSSH*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.ssh.com/ssh/openssh>>. Citado 2 vezes nas páginas 55 e 56.

YLONEN, T. *SSH*. 2020. Acesso em: 16 de nov. de 2020. Disponível em: <<https://www.ssh.com>>. Citado 3 vezes nas páginas 51, 52 e 55.

Zehl, S.; Zubow, A.; Wolisz, A. Practical distributed channel assignment in home wi-fi networks. In: *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. [S.l.: s.n.], 2017. p. 1–4. Citado 2 vezes nas páginas 34 e 40.

Zhou, M. et al. Evaluation of the node importance in power grid communication network and analysis of node risk. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2018. p. 1–5. Citado na página 32.

APÊNDICE A – *script* de instalação do pacote de dependências do *netdata*

lstinputlistingcurl.sh

APÊNDICEAPÊNDICE B

APÊNDICE B – *script Git* para copiar diretório do *netdata*

```
1 # Download do NETDATA em FEDORA
2
3 # Atualizar dependencias do sistema
4 sudo yum update
5
6 # Dependencias adicionais Git
7 sudo yum styleinstall asciidoc xmlto docbook2X getopt
8
9 # Instalar comando Git
10 sudo yum styleinstall git-all
11
12 # Descarregar arquivos – pasta 'netdata' criado em /home/lps
13 git clone https://github.com/netdata/netdata.git --depth=100
14 cd netdata
15
16 # Executar script em /home/lps/netdata/netdata-installer.sh
17 sudo ./netdata-installer.sh
```

APÊNDICEAPÊNDICE C

APÊNDICE C – *Script* de inicialização automática

```
1 # copiar dados de inicializa o
2 sudo cp /usr/sbin/netdata /etc/init.d/netdata
3
4 # atualizar
5 sudo update-rc.d netdata defaults
```

APÊNDICEAPÊNDICE D

APÊNDICE D – Sintaxe para adicionar *MGUID* no *proxy*

```
1 [MACHINE_GUID]
2     enabled = yes
3     history = 3600
4     memory mode = dbengine
5     health enabled = yes
6     allow from = *
```

APÊNDICEAPÊNDICE E

APÊNDICE E – Alterações nos arquivos *.conf* para o nó principal

```

1  # NO ARQUIVO DE CONFIGURACAO netdata.conf
2  # originalmente:
3  [global]
4      memory mode = none
5  [web]
6      mode = none
7
8  #alterado para:
9  [global]
10     memory mode = dbengine
11  [web]
12     mode = static-threaded
13
14
15  # NO ARQUIVO DE CONFIGURACAO stream.conf
16  # originalmente:
17  #2. ON MASTER NETDATA – THE ONE THAT WILL BE RECEIVING METRICS
18  #[API_KEY] is [YOUR-API-KEY], i.e [11111111-2222-3333-4444-555555555555]
19  [my-api-key]
20     enabled = no
21     allow from = *
22     default history = 86400
23     default memory mode = none
24     health enabled by default = auto
25     default postpone alarms on connect seconds = 60
26     multiple connections = allow
27
28  #GUID
29  [MACHINE_GUID]
30     # enable this host: yes | no
31     # When disabled, the master will not receive metrics for this host.
32     enabled = no
33
34  #alterado para:
35  [34319e67-1852-46ba-bd8c-6cf6ce6f2c0e]
36     enabled = yes
37     allow from = *
38     default history = 86400
39     default memory mode = none
40     health enabled by default = auto

```

```
41     default postpone alarms on connect seconds = 60
42     multiple connections = allow
43
44     #GUID
45     [valor numerico MGUID do no coletor ]
46     enabled = yes
47     history = 3600
48     memory mode = save
49     health enabled = yes
50     allow from = *
```

APÊNDICEAPÊNDICE F

APÊNDICE F – Alterações nos arquivos *.conf* para os nós secundários

```

1  # NO ARQUIVO DE CONFIGURACAO netdata.conf
2  # originalmente:
3  [global]
4      memory mode = none
5  [web]
6      mode = none
7
8  # alterado para:
9  [global]
10     memory mode = save
11 [web]
12     mode = static-threaded
13
14 # NO ARQUIVO DE CONFIGURACAO stream.conf
15 # originalmente:
16 #1. ON SLAVE NETDATA – THE ONE THAT WILL BE SENDING METRIC
17 [stream]
18     # Enable this on slaves , to have them send metrics.
19     enabled = no
20
21     # Where is the receiving netdata?
22     # A space separated list of:
23     #
24     #      [PROTOCOL:]HOST[%INTERFACE][:PORT][:SSL]
25     #
26     # If many are given , the first available will get the metrics.
27     #
28     # PROTOCOL = tcp , udp , or unix (only tcp and unix are supported by
29     #           masters)
30     # HOST      = an IPv4 , IPv6 IP , or a hostname , or a unix domain socket
31     #           path.
32     #           IPv6 IPs should be given with brackets [ip:address]
33     # INTERFACE = the network interface to use (only for IPv6)
34     # PORT      = the port number or service name (/etc/services)
35     # SSL       = when this word appear at the end of the destination
36     #           string
37     #           the Netdata will do encrypt connection with the master.
38     #
39     # This communication is not HTTP (it cannot be proxied by web proxies).
40     destination =

```

```

38
39
40 # The API_KEY to use (as the sender)
41 api key =
42
43
44 #alterado para:
45
46 [stream]
47 # Enable this on slaves , to have them send metrics.
48 enabled = yes
49
50 # Where is the receiving netdata?
51 # A space separated list of:
52 #
53 #      [PROTOCOL:]HOST[%INTERFACE][:PORT][:SSL]
54 #
55 # If many are given , the first available will get the metrics.
56 #
57 # PROTOCOL = tcp, udp, or unix (only tcp and unix are supported by
58 #           masters)
59 # HOST      = A.B.C.D
60 # INTERFACE = the network interface to use (only for IPv6)
61 # PORT      = 19999
62 # SSL       = when this word appear at the end of the destination
63 #           string
64 #           the Netdata will do encrypt connection with the master.
65 #
66 # This communication is not HTTP (it cannot be proxied by web proxies).
67 destination = A.B.C.D:19999
68
69 # The API_KEY to use (as the sender)
70 api key = 34319e67-1852-46ba-bd8c-6cf6ce6f2c0e

```

APÊNDICEAPÊNDICE G

APÊNDICE G – *Script* de alteração nós secundários

```
1 #Configurar Stream Secundario
2
3 sudo cat >/etc/netdata/netdata.conf <<EOF
4 [global]
5     memory mode = dbengine
6 [web]
7     mode = static-threaded
8 EOF
9
10 sudo cat >/etc/netdata/stream.conf <<EOF
11 [stream]
12     enabled = yes
13     destination = http://A.B.C.D:1999
14     api key = 34319e67-1852-46ba-bd8c-6cf6ce6f2c0e
15 EOF
```

APÊNDICEAPÊNDICE H

APÊNDICE H – Exemplo de modificação na *dashboard* principal

```

1
2 netdataDashboard.menu = {
3   'system': {
4     title: 'System Overview',
5     icon: '<i class="fas fa-bookmark"></i>',
6     info: 'Overview of the key system metrics.'
7   },
8
9   #[web]
10  custom_dashboard_info.js = dashboard_info_custom_example.js
11
12  # modificado para
13
14  customDashboard.menu = {
15    system: {
16      title: "Cluster LPS",
17      icon: '<i class="fa fa-university" aria-hidden="true"></i>',
18      info: "Monitoramento Cluster – Laboratorio Processamento de Sinais"
19    }
20  };
21
22  #[web]
23  custom_dashboard_info.js = lps_dashboard_info_file.js

```

APÊNDICEAPÊNDICE I

APÊNDICE I – Alterações no trecho do arquivo *.html* para o nó principal

```

1  # Alteracao para o localhost como parametro objeto Dash
2      var dash = new Dash( 'http://A.B.C.D:19999' );
3
4      var picknsort = new PickNSort(true);
5
6  # Importar dashboard.js
7  $.getScript(dash.base_url + '/dashboard.js', function() {
8      console.log("Loaded dashboard.js");
9      setTimeout(function () {
10         $('#alarms').css("visibility", "visible");
11     }, 400);
12 });
13
14 setInterval(function () {
15     dash.digest();
16 }, 6 * 1000);

```

APÊNDICEAPÊNDICE J

APÊNDICE J – A sintaxe *dash*-* exemplificada

```

1 <div class="dash-graph"           # Usar a classe dash-graph
2 data-dash-netdata="system.cpu"    # Usar data-dash-netdata para definir
    tipo dado
3 data-dygraph-valuerange="[0, 100]"> # Adicional: inclusao de qualquer
    outra data-*

4 </div>

5 <div class="dash-chart"           # Usar dash-chart para graficos tipo
    pie chart
6 data-dash-netdata="system.io"    # Usar data-dash-netdata para definir
    tipo dado
7 data-dimensions="in"             # Dimensao que sobrescreve definicao
    padrao
8 data-title="Disk Read"           # Titulo que sobrescreve definicao
    padrao
9 data-common-units="dash.io">    # Unidade que sobrescreve definicao
    padrao
10 </div>

```
